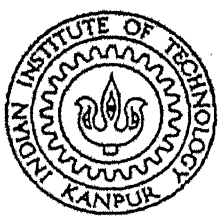
Approximation Algorithms for 3-D Common Substructu Identification in Drug and Protein Molecules

by
Samarjit Chakraborty



Department of Computer Science & Engineering
Indian Institute of Technology, Kanpur
June, 1998

Approximation Algorithms for 3-D Common Substructure Identification in Drug and Protein Molecules

A Thesis Submitted
in Partial Fulfillment of the Requirements
for the Degree of
Master of Technology

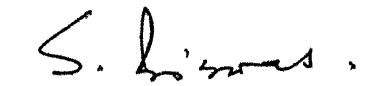
by
Samarjit Chakraborty

to the
Department of Computer Science & Engineering
Indian Institute of Technology, Kanpur
June, 1998

CSE-1998-CHA-M-APP

Certificate

Certified that the work contained in the thesis entitled "*Approximation Algorithms for 3-D Common Substructure Identification in Drug and Protein Molecules*", by Mr.*Samarjit Chakraborty*, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.



(Somenath Biswas)
Professor,
Department of Computer Science & Engineering,
Indian Institute of Technology Kanpur,
Kanpur, India.

June, 1998

Abstract

Identifying the common 3-D substructure in two drug or protein molecules is a important problem in synthetic drug design and molecular biology. This proble can be represented by the following geometric pattern matching problem : give two point sets A and B in three-dimensions, and a real number $\epsilon > 0$, find th maximum cardinality subset $S \subseteq A$ and an isometry $\mathcal{I}$, such that each point $\mathcal{I}(S)$ is within ϵ distance of a distinct point of B. Since it is difficult to sol this problem exactly, in this thesis we have theoretically studied this problem an proposed several approximation algorithms with guaranteed approximation ratio.

Our algorithms can be classified into two groups. In the first we extend th concept of approximate decision algorithms for approximate congruence of poi sets in 2-D, to approximately maximize the size of S. All the algorithms in th class exactly satisfy the constraint imposed by ϵ. In the second class of algorithn this constraint is satisfied only approximately. In this case, we improve the existir approximation ratio for this class of algorithms, while keeping the time complexi unchanged. For the existing approximation ratio, we also propose algorithms wi substantially better running time. We also suggest several improvements of the bas algorithms, all of which have a running time of $O(n^{8.5})$. These improvements main consist of using randomization, and exploiting some general structural properties protein molecules. As a result, our final algorithm has a running time of $O(n^{2.5} \log n$

Acknowledgments

I am indebted to my thesis supervisor Dr. Somenath Biswas for his guidance, and for giving a patient listening to my half baked ideas whenever I had requested for it. I would also like to thank him for helping me to learn, atleast with partial success, how to write technical documents. I am grateful to my parents, specially my father, for allowing me to attend IIT Kanpur once again and for supporting me in every possible way during the last two years of my stay here.

A lot of people are directly and as well as indirectly responsible for the successful completion of this thesis. I thank Suresh Venkatasubramanian for helping me to get introduced to this interesting field of geometric pattern matching, and for providing me the initial set of references. I would also like to thank Dr. Tatsuya Akutsu and Dr. Laurence Boxer for sending me some of their papers, Dr. Asish Mukhopadhaya for lending me some of his books and papers, and S.V. Rao for pointing out his work on the n-circle problem. The brief but stimulating discussions with Dr. N. Sathyamurthy and Dr. J. Iqbal were highly educative. I am grateful to Dr. Kalyanmoy Deb and all the members of KanGAL, specially Rajiv Mehrotra, for allowing me an unrestricted access of the facilities in their lab.

I am sincerely thankful to all those who have helped me during my stay in Kanpur. I cannot list everyone. My association with Atanu Saha helped me during the periods of emotional distress, and my acquaintance with Goutam Chakraborty has greatly enriched me with a proper attitude towards meaningful research. Life would never have been the same without Ritesh Pila, Jay Kolhatkar, Kousik Panda, Indranil Chakraborty and Jyoti Prokash Nandy. I will cherish the memory of the days spent with them for decades to some. I would finally like to thank all my classmates of M.Tech.'96, specially Himadri Sekhar Paul, Atul Kumar Dhobal, Kshitiz Krishna, K. Muralikrishna, and K.S. Shyamsunder for their help and support.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

Determining the structural similarities between a set of molecules is a central issue in both synthetic drug design and in studying biomolecular recognition and interaction of proteins. The need to solve this problem arises because of two assumptions in computational chemistry - (i) drug activity is obtained through the molecular recognition and binding of one molecule, called the *ligand*, to the pocket of another molecule, called the *receptor* [Boy90] (ii) functional properties of proteins depend on some typical parts of their three dimensional structure, called 3-D structural or *binding motifs* [BT91]. Thus two drug molecules having a large common substructure might show similar drug activity because of the substructure binding or *docking* itself into the pocket of the receptor. On the other hand, since families of proteins retain a common underlying 3-D structure, identification of this is crucial in understanding the mechanism by which they work. The receptor-ligand recognition and binding is also encountered in a large number of other biological processes, such as cell-cell recognition, enzyme catalysis (and inhibition), and in regulation of development and growth.

Again, in the case of structure-based drug design, since the geometric structures of a very few receptor molecules are known, trying to develop pharmaceutical drugs

for these receptors is an important research problem [BGSS, MBD+93]. Finding out the common substructure that is contained in some active conformation of each of the ligands, that have been experimentally found to interact with the considered receptor, can give important insight into the structure of the receptor. Since the common substructure probably docks into a pocket of the receptor, the shape of the pocket is the geometric complement of the substructure, a determination of which can lead to the design of more effective pharmaceutical drugs.

Other applications of this problem are in (i) molecular database screening, where we might want to detect a structural pattern in a large set of structures, or retrieve functionally equivalent, but structurally novel molecules from the database [Wil95], and (ii) in suggesting alignments of molecules for input to CoMFA (Comparative Molecular Field Analysis) and other 3-D QSAR (Quantitative Structure-Activity Relationship) methods [BGSS, ITY+].

1.2 The Common Substructure Identification Problem

Considering the applications detailed in the last section, we address the following problem : given two molecules, find the maximal common *rigid sub-unit* contained in both the molecules. This is equivalent to determining the rotation and translation of one molecule, relative to the other, such that there is a large *fit* or superimposition between the two molecules. Once such a common substructure is isolated, it can be used to detect its presence in other molecules exibiting similar activity, and can also be chemically evaluated to identify the degree to which it is responsible for the activity in question.

In the receptor-ligand binding problem, the receptor molecules mostly being proteins are fairly large. The ligands when proteins, are of the order of several thousand atoms, and when chemicals or drug molecules, are in the range of ten to a few hundred atoms. Thus to systematically deal with such large volumes of spatial data, the need for efficient algorithms comes into picture. Although there is a considerable volume of literature which address this problem, they still do not

seem to be sufficient from a viewpoint of computation time [Aku96], and hence leave scope for further research. More importantly, almost none of the known methods provide a theoretical guarantee for the quality of the obtained results [Aku96]. So we study this problem from a theoretical point of view and propose algorithms which have guaranteed performance bounds in terms of the quality of the results. Towards this end, we view this maximal common substructure identification problem in an abstract geometric setting where each atom is considered to be a point in 3-D space. Thus a molecule is treated as set of labeled points in $\Re^3$. This reduces to finding the point set of maximal cardinality that is contained in both the given point sets, representing the two molecules. Now, since atom positions are fuzzy, it is impractical to consider an exact match between two points. Hence considerable tolerance is allowed and two points p_1 and p_2 are said to *match* whenever $|p_1 - p_2| \leq \epsilon$, where ϵ is a predefined constant, usually referred to as *point location error.*

A protein may be viewed as a sequence (linear chain) of amino acids which folds to generate a compact domain with a specific three-dimensional structure. Many substructure matching algorithms for proteins, making use of this sequence property, are based on character string comparison algorithms. However such algorithms can hardly find useful substructures, whose nature is intrinsically 3-D [PA94, FNW92]. So we make no assumption about the linear ordering of amino acids in a protein molecule. Again, although from a chemistry point of view, the problem with drug molecules might be different from that of proteins, from a geometric standpoint both the problems are similar, except for the fact that protein molecules are usually much larger than drug molecules. Since in this thesis we will view the problem only from a geometric point of view, henceforth we will not distinguish between the two cases and will be concerned only with the abstract problem, except in Chapter 4 where we suggest some improvements in our algorithms, making use of properties specific to protein molecules..

1.3 Geometric Pattern Matching

Geometric pattern matching is a well studied problem in computational geometry, having applications in computer vision [HH92] and computer graphics. In the simplest case, given two finite point sets P and Q, it is required to be found if they are congruent, i.e. if there exists a transformation $\mathcal{T}$ consisting of translation, rotation, and possibly reflection, such that $\mathcal{T}(P) = Q$. $O(n \log n)$ algorithms to solve this problem in 3-D were given by Atkinson [Atk87] and also by Akutsu [Aku92]. For any $d \geq 3$ and point sets in $\Re^d$, an $O(n^{d-2} \log n)$ algorithm was given by Alt *et al.* [AMWW88]. A generalization of this, called the *congruent copy detection* or CCD, asks if point set Q is congruent to some subset of P [CGH+93, GMO94, dRL95]. However, the abstract problem that we formulated in the last section is more similar to a further generalization of the CCD, called the LCP or the *largest common point set problem* [ATT97, AH]. In this we ask for a set R of maximum cardinality that is congruent to some subset of P and to some subset of Q at the same time. Such an R is called the *largest common point set* between P and Q. With $|P| = n$, $|Q| = m$ and $n \geq m$, an $O((n^{1.43}m^{1.77} + n^2) \log n)$ algorithm for LCP in 2-D and an $O((\min\{n^{1.8}m^3 + n^3,\ n^{4.7}\}) \log n)$ algorithm for LCP in 3-D were given in [ATT97]. The problem of finding LCP with congruence replaced by similarity was considered by Irani and Raghavan in [IR96].

In all the above cases, input data is assumed to be exact, free from any noise or inaccuracy; and thus only *exact* matching between two points was considered. However, since such an assumption is not realistic in practice, the problem of *approximate congruence* or *ϵ-congruence* was formulated by Alt *et al.* [AMWW88]. Under this formulation two points are said to *match* whenever they are within ϵ distance of each other. All the problems mentioned in the last paragraph have their corresponding analogs under *ϵ-congruence*. Given two 2-D point sets, an algorithm to decide wheather they are ϵ-congruent under general isometry with Eucledian metric, was given in [AMWW88], with a running time of $O(n^8)$ (in contrast to the $O(n \log n)$ algorithms [AMWW88, Ata84] for testing exact congruence). This decision problem is easier if the correspondence between the matching points is known in advance. Alt *et al.* [AMWW88] and Iwanowski [Iwa90] gave $O(n \log n)$ algorithms

for some restricted decision problems and Imai *et al.* [ISI89] gave $O(n^3 \log n)$ algorithms for the general case with Eucledian metric and $O(n)$ algorithm for maximum metric. Similarly it was shown by Arkin *et al.* [AKM+92] that improved running times can be obtained if the ϵ-*balls* around each point are disjoint.

A related optimization problem is that of finding the minimum tolerance value ϵ, such that the two point sets are ϵ-congruent [AMWW88, GMO94]. A similar problem which has been extensively studied is that of finding the isometric transformation that minimize either the directed or undirected Hausdorff distance between two point sets [CGH+93, CK92, HK90, HKK92, HKS91, Rot91].

However in our case, which is thus finding the LCP of two point sets under ϵ-congruence, the value of ϵ is fixed and a point of one set is mapped to only one point of the other set, i.e. we require a bijection. This is not necessarily true in the case of measuring resemblance by the Hausdorff distance, where the mapping is defined by associating a point with its closest neighbor in the other set.

1.4 Approximation Algorithms for Approximate Congruence

Almost all algorithms for approximate congruence, except for the very trivial cases, suffer from high running times, even in two-dimensions. Indeed, Rucklidge [Ruc93] gives evidence that such methods must have high running times. This motivates the need for approximation algorithms for approximate congruence, which might have better time complexities. Note that the term approximate is used twice with two different senses, the first means that the algorithms are not exact in terms of the solutions they output, and the second refers to a property of the point sets.

1.4.1 Two Approaches for Designing Algorithms for Approximate Congruence

There are two main approaches towards this direction. The first is to specifically design an approximation algorithm, taking approximate congruence into consideration. The second is to take an algorithm for the exact matching model and use its approximate version, where approximate equality is used instead of exact equality, to

test matches. Although there is no marked distinction between the two approaches, in the available literature mostly the second approach is taken and is also probably preferred in practice. However as already mentioned, it is difficult to bound the performance of such algorithms. Work that falls into this category include that due to Irani and Raghavan [IR96] and also the recent paper by Finn *et al.* [FKL+97]. Exceptions to this approach is that of Goodrich *et al.* [GMO94] and also the work due to Schirra [Sch92, Sch88], Heffernan [Hef91] and Akutsu [Aku96].

1.4.2 Algorithms with Guaranteed Performance Bounds

As mentioned in Section 1.2, we theoretically study the common substructure identification problem, which with our present formulation is to find the LCP of two 3-D point sets under ϵ-congruence, where ϵ is the given point location error. Towards this goal we propose approximation algorithms which fall into the first of the two categories mentioned in the last subsection. The methods which we use for designing these algorithms are based largely on existing techniques (mainly for 2-D point sets of equal cardinality, and for testing the congruity of the entire point set), with a number of adaptations for our particular problem. Our problem is a generalization of the problem of testing the ϵ-congruence of two point sets in 2-D, both in terms of the dimension of the point sets and in terms of the conditions for congruity. Hence it is atleast as hard as that problem, and most likely much harder. An $O(n^8)$ algorithm for testing the ϵ-congruence of two 2-D point sets was given by Alt *et al.* [AMWW88]. The approximation algorithms that we propose have time complexity close to this.

- *(α, β)-Approximate Algorithms*

One class of approximation algorithms for testing ϵ-congruence, developed by Schirra [Sch92, Sch88], Heffernan [Hef91] and Heffernan and Schirra [HS92] solves the decision problem for values of ϵ not too close to ϵ_{opt}, where ϵ_{opt} is the smallest tolerance value for which the two point sets are ϵ-congruent. If ϵ is too close to ϵ_{opt} then the algorithm *gives up* and returns *DON'T KNOW*, otherwise it solves the problem

correctly. These algorithms are called *approximate*, since they are not always able to solve the problem. For equal cardinality point sets, an algorithm is called an (α, β)-approximate algorithm for approximate congruence if it solves the decision problem for all values of ϵ outside the interval $[\epsilon_{opt} - \alpha, \epsilon_{opt} + \beta)$.

Violating the ϵ-Constraint

Considering the optimization problem of finding the minimum tolerance value of ϵ such that two point sets are ϵ-congruent, Goodrich *et al.* [GMO94] gave an approximation algorithm which finds a rigid transformation, which enables $\delta\epsilon_{opt}$-congruence between the two point sets, where ϵ_{opt} is the required minimum value and $\delta > 1$. For 3-D point sets and general isometry, $\delta = 8$.

Although it is not clear how to formulate the LCP problem in this model, since now there are two quantities to optimize (the subset size and the distance between the individual points) [ATT97], Akutsu [Aku96] was able to make use of this technique to obtain approximation algorithms for the protein structure alignment problem. In terms of our problem, i.e. finding the LCP of two 3-D point sets under ϵ-congruence, this amounts to satisfying the ϵ-constraint *approximately*. His algorithm returns a subset of size larger than the largest common point set that can be obtained under ϵ-congruence. However, each point of such a subset is atmost 8ϵ distance away from the corresponding point in the other set.

To remedy this situation, a technique was proposed by which the factor of 8 can be reduced to any arbitrary constant $\delta > 1$, with only increasing the time complexity by a constant factor. However, the resulting algorithm is not practical because the constant factor depending on δ is very large, $O(\frac{1}{(\delta-1)^9})$.

Our Work

Our approximation algorithms are based on the last two classes of algorithms we have just mentioned, and are motivated by the papers due to Akutsu [ATT97], Schirra [Sch92] and Goodrich *et al.* [GMO94]. The factor 8 in Akutsu's algorithm being quite large, may not be tolerable in practical situations. We give an algorithm

where this factor is reduced to 2, without incurring any large constant factors. Next, instead of approximating the constraint imposed by ϵ, we propose algorithms which approximate the size of the largest common point set, and give upper and lower bounds on its size. Our algorithms are based on the ideas introduced by Schirra [Sch92], which were recently also used by Dehne *et al.* [DG97] to solve the unrestricted point set stereo matching problem. However they are nontrivial generalizations of [Sch92], because Schirra's algorithms are for solving the ϵ-congruence decision problem for two equal cardinality point sets and are thus able to make use of the centroids of the point sets. Our approximation algorithms for finding the LCP have subtle but important important differences with [Sch92] because they involve an optimization aspect and moreover cannot make use of the centroids of the point sets.

We also show that making use of randomization can reduce the time complexities to a large extent. This technique is now fairly standard for this class of problems and was recently also used by Finn *et al.* [FKL+97] for the common substructure identification problem for drug molecules. However as already mentioned in Section 1.4.1, their approach is not systematic and does not give any performance guarantee on the size of the common subset. Lastly we show that when the molecules whose substructure to be identified are protein chains, then by exploiting some general structural properties of proteins, the time complexity of our resulting algorithms reduce by a considerable extent.

1.5 Organization of the Thesis

In the next chapter we briefly outline the important approaches which are popularly used for the substructure identification problem and review related work. In Chapter 3, we formally state our abstract geometric problem and give various approximation algorithms. Chapter 4 is concerned with improvements of these algorithms, making use of randomization, the underlying geometric nature of the point sets, and the structural properties of protein molecules. Chapter 5 contains conclusions and directions for further research.

Chapter 2

Basic Approaches

As we have mentioned in the last chapter, all algorithms for the approximate congruence problem tend to have high running time. For example, the algorithm to test the ϵ-congruence of two point sets in $\Re^2$ has a running time of $O(n^8)$ [AMWW88]. Further, such algorithms are difficult to implement, as they need to compute intersections of complex algebraic surfaces [IMV]. On the other hand, from the viewpoint of practical implementation, algorithms for exact congruence with approximate equality instead of exact equality to test matches, perform better in terms of ease of implementation and running time [IMV, ATT97]. Hence this is the preferred approach in practice.

In this chapter we outline three main approaches popularly used for solving the common substructure identification problem. These are alignment [HU90], voting [ATT97], and geometric hashing [Wol90]. Although actually formulated for the exact matching model, all of them, specially the last two, for all practical purpose can be used for real world data. In all the cases we mention both deterministic and randomized algorithms based on these schemes. Finally we briefly review related work in this field, mostly those pertaining to the protein structure alignment problem and which have made use of the structural properties of the molecules in some form or the other. All the algorithms presented here are for point sets in 3-D space.

2.1 Alignment

Given two point sets P and Q, in the deterministic version of the alignment scheme, for every triplet of points (p_1, p_2, p_3) from P, all triplets (q_1, q_2, q_3) of Q are found such that the triangle $\triangle_P = (p_1, p_2, p_3)$ is congruent to the triangle $\triangle_Q = (q_1, q_2, q_3)$. Now for each $\triangle_Q$, the transformation $\mathcal{T}_{PQ}$ is calculated so that $\mathcal{T}_{PQ}(\triangle_P) = \triangle_Q$. This transformation $\mathcal{T}_{PQ}$ is now applied to the entire set P and the size of the intersection $\mathcal{T}_{PQ}(P) \cap Q$ is counted. The transformation for which the size of this intersection is maximized is the required transformation and gives the largest common point set that is contained in both P and Q. Let us denote it by $LCP(P, Q)$.

There are a number of ways in which we can reduce the running time of this algorithm, however, at the cost of a small failure probability, using standard random sampling procedures. Firstly, instead of using the entire point set P, we can randomly sample a subset R of P and use all possible triplets of R, instead of P, to calculate the transformations. If R contains atleast three points of $LCP(P, Q)$ then this procedure successfully finds it, with running time less than that required in the previous case. To further reduce the running time, we randomly sample a subset R of P and a subset S of Q. For all triplets of points in R and all congruent triplets of points in S, the corresponding transformation is calculated. This transformation is now applied to the entire point set P, to find out the number of matching points in Q. If $|LCP(P, Q) \cap R \cap \mathcal{T}^{-1}(S)| \geq 3$, where $\mathcal{T}$ is the transformation for which $\mathcal{T}(P) \cap Q$ is maximized, then this scheme successfully finds $LCP(P, Q)$. For this the sizes of R and S are so chosen that $|LCP(P, Q) \cap R \cap \mathcal{T}^{-1}(S)|$ is atleast three with high probability.

To extend this algorithm to incorporate ϵ-congruence, two points are said to match whenever they are within ϵ distance of each other. In this case, for every triangle $\triangle_P = (p_1, p_2, p_3)$, all triangles $\triangle_Q = (q_1, q_2, q_3)$ are considered for which $d(p_i, q_i) \leq \epsilon$, $(i = 1, 2, 3)$. For calculating the transformation $\mathcal{T}_{PQ}$, since there can be numerous transformations which map a point p_i into the ϵ-ball around the point q_i, usually some heuristic is used to resolve the situation. This transformation is as usual applied to the entire point set P to calculate the number of matching points in Q. The transformation which yields the maximum number of matching points

is used to calculate $LCP(P,Q)$. Now, because of the heuristic that was used to compute the transformation, it becomes difficult to give a theoretical guarantee of the size of the common point set obtained, with respect to the actual result. However in a practical situation it suffices to use this scheme.

2.2 Voting

For pairs of points $(p_1,p_2) \in P \times P$ and $(q_1,q_2) \in Q \times Q$, with $d(p_1,p_2) = d(q_1,q_2)$, and any transformation $\mathcal{T}$, let $mult_{p_1p_2,q_1q_2}(\mathcal{T})$ be the number of pairs $x \in P$ and $y \in Q$ satisfying $\mathcal{T}(\triangle_P) = \triangle_Q$ where $\triangle_P$ and $\triangle_Q$ are the triangles (p_1,p_2,x) and (q_1,q_2,y) respectively. For point sets P and Q as before, if $|\mathcal{T}(P) \cap Q| = k$, it is easy to see that $mult_{p_1p_2,q_1q_2}(\mathcal{T}) = k-2$ for any p_1 and p_2 belonging to $\mathcal{T}(P) \cap Q$ and $q_1 = \mathcal{T}^{-1}(p_1)$, $q_2 = \mathcal{T}^{-1}(p_2)$.

If $|LCP(P,Q)| = K$, then clearly $\max_{p_1p_2,q_1q_2}(\max_{\mathcal{T}} mult_{p_1p_2,q_1q_2}(\mathcal{T})) = K-2$, and the transformation achieving this maximum is the transformation that gives $LCP(P,Q)$. For pairs $(p_1,p_2) \in P \times P$ and $(q_1,q_2) \in Q \times Q$, $\max_{\mathcal{T}} mult_{p_1p_2,q_1q_2}(\mathcal{T})$ can be computed by considering all the congruent pairs of triangles $\triangle_P = (p_1,p_2,x)$ and $\triangle = (q_1,q_2,y)$, $x \in P$ and $y \in Q$, and letting each such pair cast a vote to the transformation $\mathcal{T}$ for which $\mathcal{T}(\triangle_P) = \triangle_Q$. Let us call this process *local voting* for the pairs (p_1,p_2) and (q_1,q_2). In the deterministic version of the algorithm, this local voting process is executed for all possible pairs (p_1,p_2) and (q_1,q_2), $p_i \in P, q_i \in Q$ $(i = 1,2)$, and the best result is taken. The transformation receiving the maximum number of votes is used to compute $LCP(P,Q)$.

Now since in the above deterministic algorithm, the transformation $\mathcal{T}$ receiving the largest number $(K-2)$ of votes, does so for every possible pair (p_1,p_2), $(\mathcal{T}(p_1),\mathcal{T}(p_2))$ such that $p_1,p_2 \in LCP(P,Q)$, there is a lot of redundancy in this process. This redundancy is difficult to avoid deretministically unless we know the value of K. Randomization can be used to address this problem, but again at the cost of a small failure probability, in exactly the same way as was done in the case of algorithms based on alignment. Firstly we can randomly sample a subset R of P and do a local voting for all possible pairs $(p_1,p_2) \in R \times R$ and $(q_1,q_2) \in Q \times Q$. Here

once the pairs (p_1, p_2) and (q_1, q_2) are fixed, the local voting is done exactly as in the deterministic case, scanning the entire sets P and Q (not restricting to R). Clearly if $|R \cap LCP(P,Q)| \geq 2$, this scheme successfully finds out $LCP(P,Q)$, but with a running time less than that required in the deterministic case. To further reduce the running time, we can as in the case of alignment, randomly sample subsets R of P and S of Q and do a local voting for all possible pairs $(p_1, p_2) \in R \times R$ and $(q_1, q_2) \in S \times S$. $LCP(P,Q)$ is successfully found if $|LCP(P,Q) \cap R \cap \mathcal{T}^{-1}(S)| \geq 2$, where $\mathcal{T}$ is the optimal transformation.

In the case of noisy data, incorporating *ϵ-congruence* into this voting scheme is extremely difficult and complicated. Hence such a rigid bound on the point location error is normally not imposed for practical implementation. Clearly, if there is a large common substructure between two point sets P and Q, then in the transformation space there will be a large cluster of votes around the optimal transformation $\mathcal{T}$. This is in contrast to the exact case, where all the votes corresponding to triplets of points belonging to $LCP(P,Q)$ got accumulated precisely with the transformation $\mathcal{T}$. Hence although the voting process is exactly the same as in the exact case, instead of trying to identify the transformation with the largest number of votes, the transformation space is searched for a large *cluster* of votes. Such a cluster is associated with the common substructure between P and Q.

2.3 Geometric Hashing

Given two point sets P and Q, the geometric hashing scheme divides the common point set identification problem into two phases, a preprocessing phase and a query phase. This technique is useful in cases where a point set P is compared with a large number of point sets $Q_1, Q_2, \ldots, Q_n$, to identify $LCP(P, Q_i)$ $i = 1, 2, \ldots, n$. In such a situation, the preprocessing stage is done on P, independently of any of the Q_is. The basic idea of this preprocessing is to store in a database, a representation of the point set P that is invariant under rigid transformation. By doing so, the representation of any Q_i when computed during the query phase, will present some similarity with the transformation invariant representation of P.

There can be many ways in which this transformation invariant representation is calculated. We mention here a simple approach used in [PA94]. Since three non collinear points in space specify a plane and hence a reference frame, for every three points (p_1, p_2, p_3) of P let $R_{(p_1,p_2,p_3)}$ be a reference frame defined by these three points. For each such reference frame $R_{(p_1,p_2,p_3)}$, formed by triplets of points of P, the coordinates of all other points $p \in P$ are calculated in this reference frame and these coordinates are set as an index to a hash table where the couple $(R_{(p_1,p_2,p_3)}, p)$ is stored.

During the query phase, as in the preprocessing phase, for every reference frame $R_{(q_1,q_2,q_3)}$ $(q_i \in Q,\ i = 1, 2, 3)$ the coordinates of all points $q \in Q$, in this reference frame, are calculated. These coordinates are used to retrieve the pairs $(R_{(p_1,p_2,p_3)}, p)$ from the hash table and for each such retrieved pair, a vote is casted for the triplet (p_1, p_2, p_3). Clearly, all triplets having a large number of votes belong to $LCP(P, Q)$. Randomization can again be used in the same way as was done in the case of alignment and voting. Firstly during the preprocessing phase, instead of using the entire point set P to calculate the reference frame, a randomly chosen subset R of P can be used, and the hash table be constructed. If any three points of $LCP(P, Q)$ belong to R then these points can be identified by the algorithm as they will receive a large number of votes. Moreover during the query phase also, all possible reference frames need not be considered. Depending on the sizes of the randomly sampled subsets, the running time decreases compared to the deterministic case.

2.4 Related Work

Apart from the approaches mentioned in the previous sections, algorithms based on clique detection have been studied widely in computational chemistry for this common substructure identification problem. For example DISCO [MBD+93] builds a *correspondence graph* G from two molecules. The nodes of G are all possible atom pairs, and an edge is created if pairs of atoms can be matched simultaneously. A clique detection algorithm is then used to find cliques in G. These correspond to common substructures in the two molecules. Although maximum clique detection

is NP-hard [GJ80], this algorithm works well in practice [MBD+93, Wil95]. Similar graph theoretic techniques were proposed in [MARW89] for protein alignment, where it was suggested to build an undirected, labelled and fully connected graph, whose nodes correspond to the linear representation of the secondary structure elements of the proteins (i.e. helices and strands), and the edges correspond to the angles between these elements. For finding the common substructure between two proteins, a subgraph isomorphism algorithm is used on the corresponding graphs.

The geometric hashing technique was used by Fisher *et al.* [FBNW92, FNW92] and also by Pennec *at al.* [PA94] for matching protein structures. Randomized versions of alignment were used by Finn *et al.* [FKL+97] for identifying common substructures in drug molecules and by Akutsu [Aku96] for the protein structure alignment problem. Lesk, Pascarella and Argos, and Rao and Rossmann proposed iterative improvement methods [Les91, PA92, RR73]. Vriend and Sander developed a greedy method in which small fragments of protein molecules were assembled into larger structures [VS91]. Taylor and Orengo developed a double dynamic programming technique [TO89], and Šali and Overington developed a stochastic method using probability density functions [ŠO94]. However all of these methods have one or more limitations [HOS+92] and most are not systematic, but based on some heuristic. Moreover as already mentioned, even though for practical purpose these methods may suffice in some cases, none, except the algorithms presented in reference [Aku96] give a theoretical guarantee of the quality of the output.

Chapter 3

Approximation Algorithms for 3-D Common Point Set Identification

In this chapter we first formalize the abstract geometric problem that was introduced in Chapter 1, and then give various approximation algorithms for the problem that arise out of this formalization. All of our algorithms are based on the alignment scheme described in the last chapter and have the underlying property that some aspect of the output has a guaranteed quality. There are two of these aspects that we consider here. The first is approximating the size of the largest common point set between two given point sets A and B, without violating the point location error ϵ, and the second is finding out a common subset where ϵ is satisfied only approximately, both having guaranteed approximation ratios. As mentioned in Chapter 1, the second aspect was also addressed by Akutsu [Aku96], where, if $S \subseteq A$ is the largest common point set between A and B under ϵ-congruence, then his algorithm outputs a subset $S' \subseteq A$, with $|S'| \geq |S|$ and S' being 8ϵ-congruent to some subset of B. Reducing this factor of 8 involves a large constant factor in the time complexity of the algorithm. We reduce this factor to 2 in our algorithm without incurring any such large constant factor. The approximation approach that we take here extends the ideas from [GMO94, Aku96] and also from [Sch92, Sch88] where approximate decision algorithms for approximate congruence of two equal cardinality point sets were presented.

3.1 Statement of the Problem

Definition 3.1 A point set S is ϵ-congruent to a point set S' if there exists an isometry $\mathcal{I}$ and a bijective mapping $l : S \rightarrow S'$ such that for each point $s \in S$, $d(\mathcal{I}(s), l(s)) \leq \epsilon$, where $d(\cdot, \cdot)$ is the underlying metric.

Definition 3.2 Given point sets A, B and real numbers $\epsilon > 0$ and $0 < \alpha \leq 1$, α-$LCP(A, B, \epsilon)$ is a subset S of A with $|S| \geq \alpha \min(|A|, |B|)$ such that S is ϵ-congruent to a subset of B.

This version of LCP was introduced by Finn *et al.* [FKL+97], where is was refered to as LCP-α, with the motivation that this captures the primary application quite well, since in practice the common substructure between two molecules must only be *large enough.* Clearly, for any ϵ, there exists a $\alpha_{max}(\epsilon)$ such that α-$LCP(A, B, \epsilon)$ exists for all $\alpha \leq \alpha_{max}(\epsilon)$ and for any $\alpha > \alpha_{max}(\epsilon)$, α-$LCP(A, B, \epsilon)$ does not exist. Similarly, for any $0 < \alpha \leq 1$, there exists $\epsilon_{min}(\alpha)$ such that α-$LCP(A, B, \epsilon)$ exists for all $\epsilon \geq \epsilon_{min}(\alpha)$ and for any $\epsilon < \epsilon_{min}(\alpha)$, α-$LCP(A, B, \epsilon)$ does not exist. Considering this, we address the following problem :

Input : 3-D point sets A, B and a real number $\epsilon \geq 0$
Output : $\alpha_{max}(\epsilon)$-$LCP(A, B, \epsilon)$

Unless otherwise mentioned, from now onwards a point set refers to a point set in 3-D, and any isometric transformation is a composition of just rotation and translation, not including mirror image. This restricted definition of an isometry does not result in any loss of generality, because isometry including mirror image just increases the computation time of any of our algorithm by only a constant factor. For any two point sets A and B, now the algorithm has to be run once with A and B and then with A and a mirror image of B on a plane that can be chosen arbitrarily [Aku96].

3.2 Approximating the Size of the Largest Common Point Set

In this section we state an algorithm which, when given point sets A, B and a real number $\epsilon \geq 0$, outputs a point set α_l-$LCP(A, B, \epsilon)$ and a real number α_u such that $\alpha_l \leq \alpha_{max}(\epsilon) < \alpha_u$.

For point sets A and B, let $\mathcal{I}$ be the isometry and l be the bijective mapping corresponding to α-$LCP(A, B, \epsilon_{min}(\alpha)) \subseteq A$. Let p_1 be any point of α-$LCP(A, B, \epsilon_{min}(\alpha))$. Let p_2 be the point belonging to α-$LCP(A, B, \epsilon_{min}(\alpha))$ which is farthest from p_1 and p_3 be the point of α-$LCP(A, B, \epsilon_{min}(\alpha))$, such that the perpendicular distance from p_3 to the line passing through p_1 and p_2 is maximized.

Now, let T_1 be the translation that takes the point p_1 to $l(p_1)$, R_1 be the rotation about the point $T_1(p_1)$, that causes $T_1(p_1)$, $T_1(p_2)$ and $l(p_2)$ to become collinear, and R_2 be the rotation about the $\overline{R_1(T_1(p_1))\, R_1(T_1(p_2))}$ axis, that causes $R_1(T_1(p_1))$, $R_1(T_1(p_2))$, $R_1(T_1(p_3))$ and $l(p_3)$ to become coplanar. Let $\mathcal{I}'$ be the isometric transformation which is the composition of T_1, R_1 and R_2, i.e. $\mathcal{I}'(p) = R_2(R_1(T_1(p)))$. Then the following lemma follows immediately from [GMO94].

Lemma 3.1 *The isometry $\mathcal{I}'$ and the bijective mapping l correspond to α-$LCP(A, B, 8\epsilon_{min}(\alpha))$ where $\mathcal{I}'$, l, A, B, α are as described above.*

Proof: If the set $\mathcal{I}(A)$ is translated such that the point p_1 becomes coincident with $l(p_1)$, then each point $p \in \mathcal{I}(\alpha$-$LCP(A, B, \epsilon_{min}(\alpha)))$ is within a distance $2\epsilon_{min}(\alpha)$ from its corresponding point $l(p) \in B$. Now if the resulting set is rotated about the point p_1 such that p_1, $\bar{p}_2$ and $l(p_2)$ become collinear, then the point p_2 moves by atmost a distance of $2\epsilon_{min}(\alpha)$. Since p_2 is the farthest point of α-$LCP(A, B, \epsilon_{min}(\alpha))$ from the centre of rotation, any point of α-$LCP(A, B, \epsilon_{min}(\alpha))$ will move by a distance of atmost $2\epsilon_{min}(\alpha)$. Finally, when the resulting set is rotated about the $\overline{p_1 p_2}$ axis such that p_1, p_2, p_3 and $l(p_3)$ become coplanar, the point p_3 moves by a distance of atmost $4\epsilon_{min}(\alpha)$. Since p_3 is the farthest point of α-$LCP(A, B, \epsilon_{min}(\alpha))$ from the $\overline{p_1 p_2}$ axis, any other point of α-$LCP(A, B, \epsilon_{min}(\alpha))$ moves by a distance of atmost $4\epsilon_{min}(\alpha)$. So, any point of α-$LCP(A, B, \epsilon_{min}(\alpha))$ was

moved by atmost $\epsilon_{min}(\alpha)$ by translation, $2\epsilon_{min}(\alpha)$ by the first rotation and $4\epsilon_{min}(\alpha)$ by the second rotation. Since this point was originally atmost a distance of $\epsilon_{min}(\alpha)$ away from $l(p)$, now it is atmost a distance of $8\epsilon_{min}(\alpha)$ away from $l(p)$. This proves the lemma. ∎

Definition 3.3 If P is a triplet of points and Q is another triplet of points, let T_{PQ} be the isometric transformation, as described above, i.e. if $P = (p_{i_1}, p_{i_2}, p_{i_3})$ and $Q = (q_{j_1}, q_{j_2}, q_{j_3})$, then $T_{PQ}(p_{i_1}) = q_{j_1}$; $T_{PQ}(p_{i_1})$, $T_{PQ}(p_{i_2})$ and q_{j_2} are collinear and $T_{PQ}(p_{i_1})$, $T_{PQ}(p_{i_2})$, $T_{PQ}(p_{i_3})$ and q_{j_3} are coplanar.

Definition 3.4 For point sets A, B, isometry $\mathcal{I} : A \rightarrow B$ and a real $\epsilon > 0$, let $G(\mathcal{I}, \epsilon, A, B)$ be a bipartite graph $(U \cup V, E)$ where U and V represent the points of A and B respectively and if $u \in U$ is the node corresponding to $a \in A$ and $v \in V$ corresponds to $b \in B$, then $E = \{(u, v) \mid d(\mathcal{I}(a), b) \leq \epsilon\}$.

Next, making use of the isometry stated in Lemma 3.1, we first give an approximate decision algorithm about the existance of α-$LCP(A, B, \epsilon)$ where point sets A, B, and real numbers α and ϵ are input to the algorithm. This decision algorithm is called *approximate*, because it can take a decision only for values of (α, ϵ) for which ϵ is not too close to $\epsilon_{min}(\alpha)$. When ϵ is too close to $\epsilon_{min}(\alpha)$, the algorithm may return DON'T KNOW and such values of ϵ are said to constitute the *indecision interval.* However, whenever the algorithm returns YES or NO, the answer is correct. This algorithm has an indecision interval equal to $[\frac{1}{8}\epsilon_{min}(\alpha), 8\epsilon_{min}(\alpha))$. Making use of this decision algorithm, we construct an algorithm which approximates $\alpha_{max}(\epsilon)$-$LCP(A, B, \epsilon)$ in the way mentioned at the begining of this section. We then calculate the running time of this algorithm and finally show that the quality or accuracy of this approximation is related to the size of the indecision interval $[\frac{1}{8}\epsilon_{min}(\alpha), 8\epsilon_{min}(\alpha))$.

Algorithm 3.1

Input : *Point sets A, B, real numbers* $\epsilon > 0$, $0 < \alpha \leq 1$.

Output : *If* α-$LCP(A, B, \epsilon)$ *exists, then YES or DON'T KNOW, else NO or*

DON'T KNOW.

for all triplets $P = (p_{i_1}, p_{i_2}, p_{i_3})$ from A
  for all triplets $Q = (q_{j_1}, q_{j_2}, q_{j_3})$ from B
  {
    compute T_{PQ};
    construct $G(T_{PQ}, \epsilon, A, B)$;
    if $G(T_{PQ}, \epsilon, A, B)$ has a matching of size greater than or equal to $\alpha \min(|A|, |B|)$ **then return** YES;
  }
decision = NO;
for all triplets $P = (p_{i_1}, p_{i_2}, p_{i_3})$ from A
  for all triplets $Q = (q_{j_1}, q_{j_2}, q_{j_3})$ from B
  {
    compute T_{PQ};
    construct $G(T_{PQ}, 8\epsilon, A, B)$;
    if $G(T_{PQ}, 8\epsilon, A, B)$ has a matching of size greater than or equal to $\alpha \min(|A|, |B|)$ **then** decision = YES;
  }
if (decision == NO) **then return** NO **else return** DON'T KNOW;

Theorem 3.2 *Algorithm 3.1 always returns the correct answer about the existance of* α-$LCP(A, B, \epsilon)$ *if*

(i) $\epsilon \geq 8\epsilon_{min}(\alpha)$, or

(ii) $\epsilon < \frac{1}{8}\epsilon_{min}(\alpha)$

and it either returns the correct answer or returns DON'T KNOW if $\frac{1}{8}\epsilon_{min}(\alpha) \leq \epsilon < 8\epsilon_{min}(\alpha)$.

Proof: If an isometry $\mathcal{I}$ and bijective mapping l correspond to α-$LCP(A, B, \epsilon)$, then $\mathcal{I}$ and l correspond to any α-$LCP(A, B, \epsilon')$ where $\epsilon' \geq \epsilon$. From Lemma 3.1, it follows that Algorithm 3.1 finds an isometry $\mathcal{I}$ and a mapping l which enables in

finding an α-$LCP(A, B, 8\epsilon_{min}(\alpha))$. Since $\mathcal{I}$ and l also correspond to α-$LCP(A, B, \epsilon)$ for any $\epsilon \geq 8\epsilon_{min}(\alpha)$, the algorithm returns YES for all $\epsilon \geq 8\epsilon_{min}(\alpha)$.

For any ϵ, the algorithm returns YES iff $G(T_{PQ}, \epsilon, A, B)$ has a matching of size greater than or equal to $\alpha \min(|A|, |B|)$. Hence all YES answers are correct.

For any $\epsilon < \frac{1}{8}\epsilon_{min}(\alpha)$, no isometry enables in finding α-$LCP(A, B, 8\epsilon)$. Hence for any $\epsilon < \frac{1}{8}\epsilon_{min}(\alpha)$, the algorithm always returns NO.

Finally, since the algorithm finds an isometry and bijective mapping corresponding to α-$LCP(A, B, 8\epsilon_{min}(\alpha))$, it can return NO only if $\epsilon < \epsilon_{min}(\alpha)$. Hence all NO answers are also correct. ∎

Lemma 3.3 *If $|A| = m$ and $|B| = n$, then Algorithm 3.1 has time complexity $O(m^4 n^4 \sqrt{m+n})$.*

Proof: There are $O(m^3)$ triplets of A and $O(n^3)$ triplets of B. Constructing the graph $G(T_{PQ}, \epsilon, A, B)$ takes time $O(mn)$. Computing the maximum matching in $G(T_{PQ}, \epsilon, A, B)$ can be done using Hopcroft and Karp's algorithm [HK73] in time $O(mn\sqrt{m+n})$. ∎

Algorithm 3.2
Input : *Point sets A, B and a real number $\epsilon > 0$.*
Output : *Real numbers $0 < \alpha_l \leq \alpha_u \leq 1$.*

$M = 0$;
for all triplets $P = (p_{i_1}, p_{i_2}, p_{i_3})$ from A
  for all triplets $Q = (q_{j_1}, q_{j_2}, q_{j_3})$ from B
  {
    compute T_{PQ};
    construct $G(T_{PQ}, \epsilon, A, B)$;
    M' = size of the maximum matching in $G(T_{PQ}, \epsilon, A, B)$;
    if $(M' > M)$ then { $M = M'$; $T = T_{PQ}$; }
  }
$\alpha_l = M/\min(|A|, |B|)$;

$M = 0$;
for all triplets $P = (p_{i_1}, p_{i_2}, p_{i_3})$ from A
  for all triplets $Q = (q_{j_1}, q_{j_2}, q_{j_3})$ from B
  {
    compute T_{PQ};
    construct $G(T_{PQ}, 8\epsilon, A, B)$;
    M' = size of the maximum matching in $G(T_{PQ}, 8\epsilon, A, B)$;
    if $(M' > M)$ **then** $M = M'$;
  }
$\alpha_u = (M+1)/\min(|A|, |B|)$;
return (α_l, α_u);

This algorithm is based on the same idea as that of the decision Algorithm 3.1. Clearly, for a fixed ϵ, α_l is the largest value of α for which Algorithm 3.1 would return YES and α_u is the smallest value of α for which Algorithm 3.1 would return NO.

Theorem 3.4 *If $|A| = m$ and $|B| = n$, then Algorithm 3.2 has time complexity $O(m^4 n^4 \sqrt{m+n})$.*

Proof: Same as that for Lemma 3.3. ∎

Theorem 3.5 *Given point sets A, B and a real number $\epsilon > 0$, Algorithm 3.2 returns real numbers $0 < \alpha_l \leq \alpha_u \leq 1$, such that $\alpha_l \leq \alpha_{max}(\epsilon) < \alpha_u$ and*

(i) $\alpha_l \geq max\{\alpha : \epsilon > 8\epsilon_{min}(\alpha)\}$

(ii) $\alpha_u \leq min\{\alpha : \epsilon < \frac{1}{8}\epsilon_{min}(\alpha)\}$

Proof: Clearly, Algorithm 3.1 returns YES for $\alpha = \alpha_l$ and NO for $\alpha = \alpha_u$. Hence the first part of the theorem is obvious, since α_l-$LCP(A, B, \epsilon)$ exists and α_u-$LCP(A, B, \epsilon)$ does not exist, along with the fact that, for all $\alpha \leq \alpha_{max}(\epsilon)$, α-$LCP(A, B, \epsilon)$ exists and for any $\alpha > \alpha_{max}(\epsilon)$, α-$LCP(A, B, \epsilon)$ does not exist.

The second part of the theorem follows from Theorem 3.2 along with the fact that if $\alpha_1 \leq \alpha_2$ then, $\epsilon_{min}(\alpha_1) \leq \epsilon_{min}(\alpha_2)$. ∎

3.3 An Exact Algorithm for Finding the Largest Common Point Set under Rotation

Now we describe an algorithm which on given point sets A, B, a real number $\epsilon > 0$, and a fixed point p, finds $\alpha_{max}(\epsilon)$-$LCP(A, B, \epsilon)$ where the underlying isometry consists of only pure rotation about the point p. We shall make use of this algorithm in the next section to decrease the indecision interval of Algorithm 3.1 and thereby obtain a better approximation of $\alpha_{max}(\epsilon)$-$LCP(A, B, \epsilon)$ under general isometry, compared to what is obtained by Algorithm 3.2.

To understand this algorithm, consider ϵ-balls around each point of the set B. As the set A is rotated about the point p, points of A move into and out of the ϵ-balls of B. The problem is essentially that of finding the rotation for which maximum number of points of A are within distinct balls of B. If A and B were 2-D point sets, then a straightforward approach could have been as follows : for each point $a_i \in A$ and each point $b_j \in B$, let θ_{ij} be the angular interval for which a_i is within the ϵ-ball around b_j, when the set A is rotated about the point p. The end points of these intervals θ_{ij} are sorted to get a new set of intervals. Now, we move through this new set and for each angular interval in this set, construct the bipartite graph as was done in Algorithms 3.1 and 3.2, in which each edge (a_i, b_j) indicates that a_i is within the ϵ-ball around b_j for this angular interval. The interval for which this graph has the largest maximum matching, results in finding the largest common point set between A and B under rotation about p. For 3-D point sets, we do not obtain angular intervals, but rather solid angles. So moving through these angles is not as straightforward as in the case of two dimensions.

To describe the algorithm we make use of the following notation. If p is the fixed point about which the point set A is rotated, then for any point $a_i \in A$, S_{a_i} denotes the sphere of radius $d(p, a_i)$ centered at the point p, and for any point $b_j \in B$, S_{b_j} denotes the sphere of radius ϵ centered at b_j, i.e. S_{b_j} denotes the ϵ-ball around the point b_j. S_p will be used to denote a sphere centered at p. The radius of this sphere is less than the distance of p from the nearest point of either A or B. This implies that no point of either A or B is within the sphere S_p. Although this condition

is not necessary for the algorithm to work, it will make the situation simpler by reducing the number of different possible cases. If spheres S_{a_i} and S_{b_j} intersect, the intersection of their surfaces will be a circle [Lei43], which we denote by C_{ij}. Consider the solid cone having p as its vertex and the circle C_{ij} as its base. This solid cone intersects with the sphere S_p, to form a circular figure called a *dome*, on the surface of S_p. This dome is indicative of the solid angle corresponding to which the point a_i is within the ϵ-ball around the point b_j. Now consider a fixed point p' on the surface of S_p. If the same set of rotations which are to be applied to the point a_i for it to lie within the ϵ-ball around b_j, are applied to the point p', then it traces out an exactly similar dome on S_p. Let us denote this dome traced out by the point p', by D_{ij} (see Figure 1).

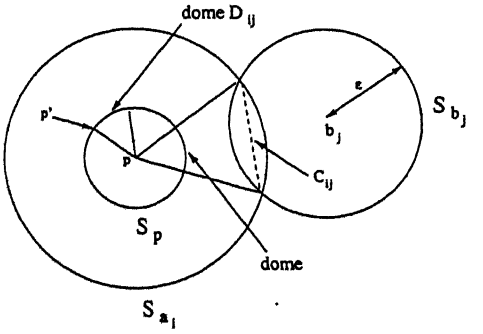


Figure 1: Dome D_{ij} resulting from points a_i and b_j

If there is a rotation R of the set A, about the point p, such that $d(R(a_i), b_j) \leq \epsilon$, then obviously $D_{ij} \neq \emptyset$. Clearly, for any point on the dome D_{ij}, there exists a rotation R such that $d(R(a_i), b_j) \leq \epsilon$. From now onwards all our references to domes are to the circular figures on the surface of S_p, traced out by the point p' as the point set A is rotated so that a point of A is within the ϵ-ball of another point of B. Now consider every point on the surface of the sphere S_p to be associated with a *membership vector*, which is indicative of all the domes to which the point belongs. Then each dome partitions the surface of S_p into two regions, and all the domes arising out of the points of A and B define a partition of the surface of S_p into a number of distinct regions, where a *region* is defined as a set of points having the same membership vector. Therefore, for any point on the region $D_{i_1 j_1} \cap D_{i_2 j_2} \cap \ldots \cap D_{i_k j_k}$, there is a rotation R such that $d(R(a_{i_l}), b_{j_l}) \leq \epsilon$, $l = 1, 2, \ldots, k$. This gives rise

to a bipartite graph, in which the nodes correspond to the points of A and B, and the edges consist of all pairs (a_{i_l}, b_{j_l}), $l = 1, 2, \ldots, k$. The region in which this graph has the largest maximum matching is our required region, because a rotation corresponding to any point in this region finds the largest common point set between A and B. To find the largest maximum matching, it is required to traverse through all the regions and find the maximum matching in the bipartite graph arising in each region. To systematically traverse through all these regions we make use of a sweep approach. Towards this, we sweep a plane $h(t) : x = t$, through the sphere S_p, starting from its leftmost end and ending in its rightmost end. At any position $x = t$, this plane intersects with the sphere S_p resulting in a circle $C(t)$. This situation is illustrated in Figure 2.

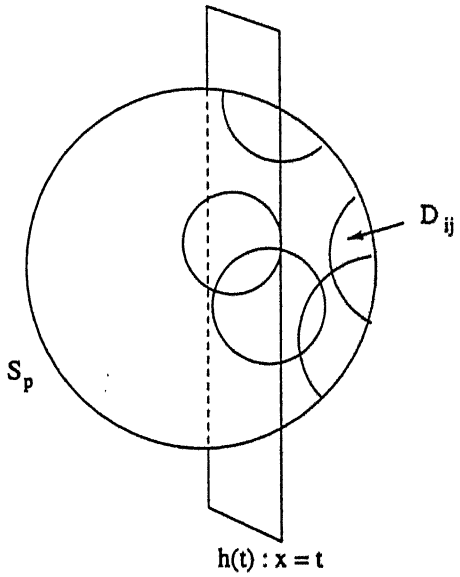


Figure 2: Sweeping the plane $h(t) : x = t$ through the sphere S_p

Any such circle $C(t)$, being a section of the surface of S_p, intersects with some of the domes on the surface of S_p. For any such dome D_{ij}, let θ_{ij} be the angle subtended by the intersection points of D_{ij} with $C(t)$, at the centre of the circle $C(t)$ (Figure 3).

For all such intersecting D_{ij}, we compute the corresponding angular intervals θ_{ij}. We next sort the end points of these angular intervals, which as a result gives rise to a new set of intervals. If an interval is contained within the domes

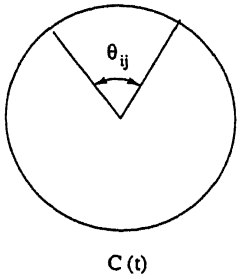


Figure 3: Angle θ_{ij} resulting out of the intersection of the dome D_{ij} with the circle $C(t)$

$D_{i_1j_1}, D_{i_2j_2}, \ldots, D_{i_kj_k}$, then it results in the bipartite graph corresponding to the region formed by the intersection of these domes. Hence if we find the maximum matching in all the graphs corresponding to each interval for this position of the plane, and repeat this process for *all* positions of the plane by sweeping it through S_p, then the largest maximum matching over all possible graphs is our required result. However, as is normal in any sweep algorithm, this (impossible) task of generating an infinite set of planes can be avoided. Let $x_1, x_2, \ldots, x_n$ be the sorted x-coordinates of all the left and right end points, and all possible intersection points, of all the domes D_{ij}. Then for any position of the plane between two consecutive x_is, i.e. $h(t) : x = t,\ t \in (x_i, x_{i+1})$ $(i = 1, 2, \ldots, n-1)$, the same set of graphs arise out of the intersection of the circle $C(t)$ with the domes lying on it. Hence it is sufficient to consider only one position of the plane between any two of these successive x_is. If A and B are point sets of cardinality m and n, then there are $O(mn)$ domes on S_p and hence $O(m^2n^2)$ intersection points. For each position of the plane $h(t) : x = t$, there are $O(mn)$ angular intervals in the corresponding circle $C(t)$. Hence this scheme needs $O(m^3n^3)$ invocations of the graph matching algorithm. However, note that $O(mn)$ domes divide the surface of the sphere S_p into $O(m^2n^2)$ regions. Hence the graph matching algorithm should be invoked only $O(m^2n^2)$ times. For this we note that if $S_1, S_2, \ldots, S_k$ are sets of domes, such that for each S_i, there is a region r_i which is contained in all the domes of S_i, then it is sufficient to do a graph matching only for the regions r_i, $i = 1, 2, \ldots, k$. Obviously the sets $S_1, S_2, \ldots, S_k$ need not be disjoint. Figure 4 illustrates this point. It is so, because, say if $D_{i_1j_1}$ and $D_{i_2j_2}$ intersect, then the graph arising out of the common region will have its edge set as

$\{(a_{i_1}, b_{j_1}), (a_{i_2}, b_{j_2})\}$, but the other two graphs will have their edge sets as $\{(a_{i_1}, b_{j_1})\}$ and $\{(a_{i_2}, b_{j_2})\}$ only. Hence the size of the maximum matching in the graph arising out of the common region will be atleast as large as the maximum matching in graphs corresponding to the two other regions.

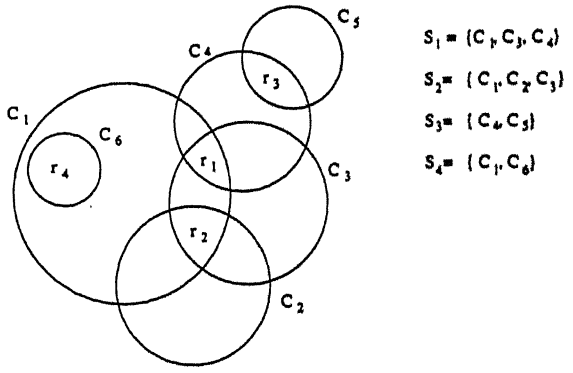


Figure 4: It is sufficient to do graph matching corresponding to r_1, r_2, r_3 and r_4 only

Therefore if $D_{i_1 j_1}, D_{i_2 j_2}, \ldots, D_{i_k j_k}$ be a group of domes such that there is a region r which is contained in all of them, our objective is to identify this region and all the domes containing it, using our sweep algorithm. Observe that there are three possible cases : i) The region r does not share its boundary with atleast one of the domes i.e. r is in the interior of the dome (Figure 5(a)). ii) One of the domes constitute the region r (Figure 5(b)). iii) r shares its boundary with all the domes (Figure 5(c)).

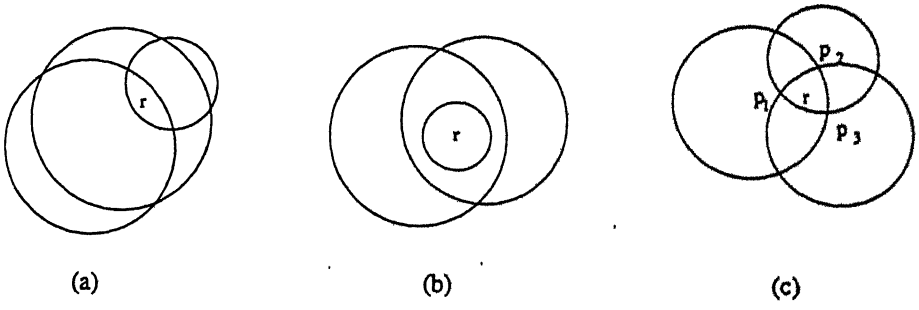


Figure 5: Three possible arrangements of domes

Let us consider the third case first. If $p_1, p_2, \ldots, p_n$ be the intersection points lying on the boundary of r, then clearly all of these points have their membership vectors equal equal to that of any point in the region r. Hence it is sufficient to do a

graph matching corresponding to any of these points. The same scheme works in the first case also, if we consider all possible intersection points. This is because all the intersection points lying on the boundary of r also lies inside the dome which does not share its boundary with r. However, this scheme of doing the graph matching at the intersection points fails in the second case. To take care of this, note that any point lying on the boundary of the dome which constitutes the region, can fulfill our requirements. For any such point, we can find out all the domes to which this point belongs and construct the corresponding graph, which will be the graph underlying r. We then do a graph matching corresponding to this graph.

Now we give the overall algorithm. Observe that the *membership vector* of the sweep plane indicating the domes which are intersected by the plane, changes only in two situations : (i) the sweep-plane just crossing the left end-point of the dome, after which this dome starts intersecting with the sweep-plane (ii) the sweep-plane just crossing the right end-point of the dome, after which this dome ceases to intersect with the sweep-plane. Our *event point schedule*, i.e. is the sequence of abscissae ordered from left to right, which define the halting positions of the sweep-plane, are made up of the x-coordinates of the left and right end points, and all the intersection points of all the domes lying on S_p.

When the event point is the left end-point of a dome, we update the membership vector of the sweep-plane to indicate that this dome now intersects with the sweep-plane. Next we construct the bipartite graph corresponding to this point making use of the information in the sweep-plane membership vector, because this point can lie only on the subset of all the domes which intersect with the sweep-plane. If the event point is contained in the domes $D_{i_1j_1}, D_{i_2j_2}, \ldots, D_{i_kj_k}$, then the nodes of the bipartite graph correspond to the points of A and B and the edges join pairs $a_{i_l} \in A$, $b_{j_l} \in B$, $l = 1, 2, \ldots, k$. We then compute the size of the maximum matching in this graph. If the event point is an intersection point of a number of domes, then we construct the bipartite graph corresponding to this point as was done in the previous case, and find the maximum matching in this graph. Finally, if the event point is the right end-point of a dome, then we just update the membership vector of the sweep-plane to indicate that from now this dome ceases to intersect with the

sweep-plane. The largest maximum matching obtained over all the graphs is our required result. Now we formally state our algorithm for finding the largest common point set under pure rotation, and the consequent theorem concerning its running time.

Algorithm 3.3

Input : *Point sets A, B, a fixed point p, and a real number $\epsilon > 0$.*

Output : *$\alpha_{max}(\epsilon)$ for pure rotation of the set A about the point p.*

```
for each a_i ∈ A
    for each b_i ∈ B
      compute D_ij;
x-structure = sorted x-coordinates of the left and right end-points and intersection
points of all the domes D_ij;
M = 0;
for each t ∈ x-structure
{
  if (t == left end-point of a dome) then
  {
    update sweep-plane membership vector;
    construct graph G corresponding to the point t using information in the sweep-plane
membership vector;
    M' = size of the maximum matching in G;
  }
  if (t == intersection point of domes) then
  {
    construct graph G corresponding to the point t using information in the sweep-plane
membership vector;
    M' = size of the maximum matching in G;
  }
  if (t == right end-point of a dome) then
    update sweep-plane membership vector;
```

if $(M' > M)$ **then** $M = M'$;
}
return $M/\min(|A|, |B|)$;

Theorem 3.6 *If* $|A| = m$ *and* $|B| = n$, *then Algorithm 3.3 has time complexity* $O(m^3n^3\sqrt{m+n})$.

Proof: The m points of the set A and n points of the set B result in $O(mn)$ domes on the surface of S_p. Hence there are $O(mn)$ end-points and $O(m^2n^2)$ intersection points. Corresponding to each of the $O(m^2n^2)$ event points, constructing the bipartite graph takes $O(mn)$ time and finding the maximum matching takes $O(mn\sqrt{m+n})$ time. Hence the overall time complexity is $O(m^3n^3\sqrt{m+n})$. ∎

3.4 Reducing the Indecision Interval

It was shown in Section 3.2 that the indecision interval of the decision Algorithm 3.1 is $[\frac{1}{8}\epsilon_{min}(\alpha), 8\epsilon_{min}(\alpha))$. In this section we shall make use of Algorithm 3.3 and reduce the indecision interval to $[\frac{1}{2}\epsilon_{min}(\alpha), 2\epsilon_{min}(\alpha))$. Using this new decision algorithm, we next give an algorithm which better approximates the size of $\alpha_{max}(\epsilon)$-$LCP(A, B, \epsilon)$ compared to Algorithm 3.2. For this we first give a lemma which is based on the same idea as that of Lemma 5 of [Sch92], but adapted for our particular problem. Here, for arbitrary points a and b in space, we use t_{ab} to denote the translation that maps a to b.

Lemma 3.7 *Let isometry* $\mathcal{I}$, *which is a composition of translation and rotation, and a bijective mapping* l, *correspond to* α-$LCP(A, B, \epsilon)$. *Let* $a \in \alpha$-$LCP(A, B, \epsilon)$. *There exists a rotation* R *about the point* $\mathcal{I}(a)$, *such that* R *and* l *correspond to* α-$LCP(t_{a\mathcal{I}(a)}(A), B, \epsilon)$. *Let* b *be an arbitrary point in space. There is a rotation* R' *about the point* b *such that* R' *and* l *correspond to* α-$LCP(t_{ab}(A), B, \epsilon + d(b, \mathcal{I}(a))$.

Proof: The first part of the lemma is obvious. From this it follows that $\mathcal{I} = R \circ t_{a\mathcal{I}(a)}$ (where $\circ$ is used to denote composition carried out from right to left). For the second part of the lemma, we show that $R' = t_{\mathcal{I}(a)b} \circ R \circ t^{-1}{}_{\mathcal{I}(a)b}$.

For any point $p \in \alpha\text{-}LCP(A, B, \epsilon)$,

$$\begin{aligned} d(R' \circ t_{ab}(p), l(p)) &= d(\, t_{\mathcal{I}(a)b} \circ R \circ t^{-1}{}_{\mathcal{I}(a)b} \circ t_{ab}(p),\ l(p)\,) \\ &= d(\, t_{\mathcal{I}(a)b} \circ R \circ t_{a\mathcal{I}(a)}(p),\ l(p)\,) \\ &= d(\, t_{\mathcal{I}(a)b} \circ \mathcal{I}(p),\ l(p)\,) \\ &\leq d(l(p), \mathcal{I}(p)) \ +\ d(\mathcal{I}(p), t_{\mathcal{I}(a)b} \circ \mathcal{I}(p)) \\ &\leq \epsilon + d(b, \mathcal{I}(a)) \end{aligned}$$

∎

Algorithm 3.4

Input : *Point sets A, B, real numbers* $\epsilon > 0$, $0 < \alpha \leq 1$.

Output : *If* α*-LCP*(A, B, ϵ) *exists, then YES or DON'T KNOW, else NO or DON'T KNOW.*

```
for each point a ∈ A
    for each point b ∈ B
    {
      α' = Algorithm 3.3(t_ab(A), B, b, ε);
      if (α' ≥ α) then return YES;
    }
decision = NO;
for each point a ∈ A
    for each point b ∈ B
    {
      α' = Algorithm 3.3(t_ab(A), B, b, 2ε);
      if (α' ≥ α) then decision = YES;
    }
if (decision = = NO) then return NO else return DON'T KNOW;
```

Theorem 3.8 *If A and B are point sets of cardinality m and n, then Algorithm 3.4 runs in time* $O(m^4 n^4 \sqrt{m+n})$ *and always returns the correct answer about the existance of* α*-LCP*(A, B, ϵ) *if*

(i) $\epsilon \geq 2\epsilon_{min}(\alpha)$, **or**

(ii) $\epsilon < \frac{1}{2}\epsilon_{min}(\alpha)$

and it either returns the correct answer or returns DON'T KNOW if $\frac{1}{2}\epsilon_{min}(\alpha) \leq \epsilon < 2\epsilon_{min}(\alpha)$.

Proof: There are $O(mn)$ invocations to Algorithm 3.3. Hence the time complexity of Algorithm 3.4 is $O(m^4n^4\sqrt{m+n})$. Let isometry $\mathcal{I}$ and a bijective mapping l, correspond to α-$LCP(A, B, \epsilon_{min}(\alpha))$. Let $a \in \alpha$-$LCP(A, B, \epsilon_{min}(\alpha))$. It follows from Lemma 3.7 that there exists a rotation R about the point $l(a)$, such that R and l correspond to α-$LCP(t_{al(a)}(A), B, \epsilon_{min}(\alpha) + d(l(a), \mathcal{I}(a)))$. Since $d(l(a), \mathcal{I}(a)) \leq \epsilon_{min}(\alpha)$, R and l correspond to α-$LCP(t_{al(a)}(A), B, 2\epsilon_{min}(\alpha))$. The rest of the proof is similar to that for Theorem 3.2. ∎

Now in the same way as was done in Section 3.2, this decision algorithm can be modified to give an approximation algorithm, which when given point sets A, B, and real number $\epsilon > 0$, outputs α_l and α_u such that $\alpha_l \leq \alpha_{max} \leq \alpha_u$, and finds the common point set α_l-$LCP(A, B, \epsilon)$. In contrast to Theorem 3.5, in this case $\alpha_l \geq \max\{\alpha : 2\epsilon_{min}(\alpha)\}$, and $\alpha_u \leq \min\{\alpha : \epsilon < \frac{1}{2}\epsilon_{min}(\alpha)\}$. Hence this gives a significantly better approximation of $\alpha_{max}(\epsilon)$-$LCP(A, B, \epsilon)$. Clearly this approximation algorithm will also have a time complexity equal to that of the decision algorithm.

3.5 Approximately Satisfying the ϵ-Constraint

In all the algorithms presented so far, the constraint imposed by the point location error ϵ was strictly satisfied. In this section we present an algorithm, which on given point sets A, B, and real number $\epsilon > 0$, instead of approximating $\alpha_{max}(\epsilon)$ as was done till now, approximately satisfies the constraint imposed by ϵ. Towards this, the algorithm outputs a subset $S \subseteq A$ which is 2ϵ-congruent to some subset of B, and $|S| \geq |\alpha_{max}(\epsilon)$-$LCP(A, B, \epsilon)|$. This approach was also followed by Akutsu [Aku96] and his algorithm outputs a subset which is 8ϵ-congruent to some subset of B. We have made use of our exact algorithm for finding $\alpha_{max}(\epsilon)$-$LCP(A, B, \epsilon)$ under pure rotation, to improve this bound to 2ϵ.

Algorithm 3.5
Input : *Point sets A, B, real number $\epsilon > 0$.*
Output : *A real number α.*

$\alpha = 0$;
for each point $a \in A$
  for each point $b \in B$
  {
    $\alpha' =$ Algorithm 3.3$(t_{ab}(A), B, b, 2\epsilon)$;
    if $(\alpha' \geq \alpha)$ then $\alpha = \alpha'$;
  }
return α;

Theorem 3.9 *Given point sets A, B, and a real number $\epsilon > 0$, if Algorithm 3.5 returns a real number α, then there exists a subset S of A with $|S| = \alpha \min(|A|, |B|)$, which is 2ϵ-congruent to some subset of B and $\alpha \geq \alpha_{max}(\epsilon)$.*

Proof: Clearly, it follows from the construction of Algorithms 3.3 and 3.5 that if Algorithm 3.5 returns α, then there exists a subset S of A with $|S| = \alpha \min(|A|, |B|)$. Let isometry $\mathcal{I}$ and bijection mapping l correspond to $\alpha_{max}(\alpha)$-$LCP(A, B, \epsilon)$. If $a \in \alpha_{max}(\epsilon)$-$LCP(A, B, \epsilon)$, then from Lemma 3.7 it follows that there exists a rotation R about the point $\mathcal{I}(a)$ such that $\mathcal{I} = R \circ t_{a\mathcal{I}(a)}$, and the isometry $\mathcal{I}' = R \circ t_{al(a)}$ correspond to α-$LCP(A, B, 2\epsilon)$, where $\alpha \geq \alpha_{max}(\epsilon)$. Since Algorithm 3.5 loops through all possible pairs of points of A and B, certainly Algorithm 3.3 gets called for a pair $a, l(a)$ where $a \in \alpha_{max}(\epsilon)$-$LCL(A, B, \epsilon)$. Hence the theorem holds. ∎

Theorem 3.10 *If $|A| = m$ and $|B| = n$, then Algorithm 3.5 has time complexity $O(m^4 n^4 \sqrt{(m+n)})$.*

Proof: There are mn calls to Algorithm 3.3. ∎

3.6 Running Time Versus Size of the Indecision Interval

The algorithm that we gave in Section 3.2 had an indecision interval of $[\frac{1}{8}\epsilon_{min}(\alpha), 8\epsilon_{min}(\alpha$ Using our exact algorithm of Section 3.3 we reduced this indecision interval to $[\frac{1}{2}\epsilon_{min}(\alpha), 2\epsilon_{min}(\alpha))$ in Section 3.4, and showed that it leads to a better approximation of $\alpha_{max}(\epsilon)$. Making use of a technique first described in [Sch92], in this section we show that this indecision interval can be made arbitrarily small, however, at the expense of increased running time. Given point sets A, B, and a real number ϵ, we can reduce the indecision interval to $[\epsilon_{min}(\alpha) - \gamma, \epsilon_{min}(\alpha) + \gamma)$, by slightly modifying Algorithm 3.4. But doing this introduces a term $(\epsilon/\gamma)^3$ in the running time of the algorithm. In this algorithm we cover the ϵ-balls around each point of the set B with balls of radius γ. Let B' be the set of points which are the centers of these γ-balls.

Algorithm 3.6
Input : *Point sets A, B, real numbers $\epsilon > 0, 0 < \alpha \leq 1; 0 < \gamma < \epsilon$.*
Output : *If α-LCP(A, B, ϵ) exists, then YES or DON'T KNOW, else NO or DON'T KNOW.*

```
for each point a ∈ A
    for each point b' ∈ B'
    {
      α' = Algorithm 3.3(t_ab'(A), B, b', ε);
      if (α' ≥ α) then return YES;
    }
decision = NO;
for each point a ∈ A
    for each point b' ∈ B'
    {
      α' = Algorithm 3.3(t_ab'(A), B, b', ε + γ);
      if (α' ≥ α) then decision = YES;
```

}
if (decision = = NO) then return NO else return DON'T KNOW;

Theorem 3.11 *Algorithm 3.6 always returns the correct answer about the existance of* α-$LCP(A, B, \epsilon)$ *if*

(i) $\epsilon \geq \epsilon_{min}(\alpha) + \gamma$, **or**

(ii) $\epsilon < \epsilon_{min}(\alpha) - \gamma$

and it either returns the correct answer or returns DON'T KNOW if $\epsilon_{min}(\alpha) - \gamma \leq \epsilon < \epsilon_{min}(\alpha) + \gamma$.

Proof: Let isometry $\mathcal{I}$ and bijective mapping l, correspond to α-$LCP(A, B, \epsilon_{min}(\alpha))$. Let $a \in \alpha$-$LCP(A, B, \epsilon_{min}(\alpha))$. Then clearly the isometry $\mathcal{I}$ maps a into the ϵ-ball around $l(a)$, and hence a γ-ball inside this ϵ-ball. Let the center of this γ-ball be b'. From Lemma 3.7 it follows that there exists a rotation R, which along with l correspond to α-$LCP(t_{ab'}(A), B, \epsilon_{min}(\alpha) + \gamma)$. The rest of the proof is exactly along similar lines as that for Theorem 3.2. ∎

Theorem 3.12 *If* $|A| = m$, $|B| = n$, *then Algorithm 3.6 has a time complexity* $O((\epsilon/\gamma)^3 m^4 n^4 \sqrt{m+n})$.

Proof: $(2\epsilon/\gamma)^3$ balls of radius γ are sufficient to cover each ϵ-ball around the points of B. So for each point of B, there are $O((\epsilon/\gamma)^3)$ additional iterations compared to that in Algorithm 3.4. ∎

In the same way, as was mentioned in Section 3.4, this improved decision algorithm leads to an even better approximation of $\alpha_{max}(\epsilon)$. However the running time of such an algorithm increases by $O((\epsilon/\gamma)^3)$. By making γ arbitrarily small, we can now get α_l and α_u which will be closer to $\alpha_{max}(\epsilon)$ compared to what was obtained in Section 3.4.

This same technique can be applied to Algorithm 3.5 to reduce the factor of 2 to any $\delta > 1$, by appropriately choosing γ. But here also we shall get a factor of

$(\epsilon/\gamma)^3$ in the running time. This is however significantly better than the algorithm proposed by Akutsu [Aku96] which obtains the same result but introduces a factor of $(\epsilon/\gamma)^9$ in the running time.

Chapter 4

Improvements Using Geometry, Randomization, and Structure of the Molecules

In this chapter we present three different modifications of the basic algorithms of the last chapter, which improve their running time. The first two are completely general, while the third relies on some structural properties of protein molecules.

4.1 Using an Approximation Algorithm for Maximum Matching in Bipartite Graphs

All the algorithms presented in the last chapter make use of the Hopcroft and Karp's algorithm [HK73] for finding the maximum matching in a bipartite graph. Clearly, any improved algorithm for maximum matching would result in improving the overall time complexity of our algorithms. It was shown by Efrat and Itai [EI96] that if the nodes of a bipartite graph are points in some d-dimensional space, and the edges are pairs of points which are within some specified distance of each other, as in our case, then atleast under certain circumstances, algorithms with better time complexities can be obtained. This was done by making use of the geometric

nature of the problem, rather than trying to solve it from a purely graph-theoretic perspective. Work related to this was also done by Vaidya [Vai89], and by Agarwal, Efrat and Sharir [AES95].

In this section we show that making use of the scheme of Efrat and Itai [EI96] along with a data structure proposed by Arya *et al.* [AMN+94] for answering nearest neighbor queries for points in $\Re^d$, we can reduce the time complexity of all our algorithms. However, the resulting algorithms have an increased indecision interval and the point location error ϵ increases by a constant factor.

4.1.1 Maximum Matching Using Geometry

For the sake of completeness, in this subsection we outline parts of Efrat and Itai's scheme that we shall make use of in our algorithm in the next subsection. Given point sets A, B, and a real number $\epsilon > 0$, consider the graph $G = (A \cup B, E)$ where E consists of all pairs (a, b) $a \in A$, $b \in B$, for which $d(a, b) \leq \epsilon$. Our problem is to find a maximum graph-matching in G.

First we recapitulate some basic definitions. A matching M of $G = (A \cup B, E)$ is a subset of E such that no vertex of G is incident on more than one edge of M. The vertices incident on M are called *matched* and the remaining are referred to as *exposed*. A path $\pi = (v_1, v_2, \ldots, v_{2t})$ is called an *alternating path* if v_1 is an exposed vertex of A, $(v_{2i}, v_{2i+1}) \in M$ and $(v_{2i-1}, v_{2i}) \in E \setminus M$ $(i = 1, 2, \ldots, t)$. Obviously the odd vertices of π belong to A and the even vertices to B. This path is called an *augmenting path* if v_{2t} is an exposed vertex. If π is an augmenting path, then $M' = (M \setminus \pi) \cup (\pi \setminus M)$ is also a matching and $|M'| = |M| + 1$. Moreover, a theorem due to Berge [Ber57] states that a matching is maximum if and only if there is no augmenting path. Hence an algorithm for maximum matching can start with an empty matching and augment it by finding augmenting paths, until none is found.

Given a matching M, Efrat and Itai's algorithm finds all shortest augmenting paths by first conducting a breadth-first search over the vertices of G to construct a layer graph $\mathcal{L}$ consisting of layers $L_1, L_2, \ldots, L_{2t}$. The first layer L_1, contains all the exposed vertices of A; L_{2i} contains all vertices of B not appearing in $\bigcup_{j<2i} L_j$, and connected through edges of G to some vertex of L_{2i-1}. If L_{2i} contains any exposed

vertices, then this is the last layer. Otherwise L_{2i+1} is constructed, which contains all vertices connected through the matching M to vertices in L_{2i}. The layer graph $\mathcal{L}$ consists of the vertex set $\bigcup_{i=1}^{2t} L_i$, edges of M that connect vertices of L_{2j} to vertices of L_{2j+1}, and edges of G that connect vertices of L_{2j-1} to vertices of L_{2j}.

Towards finding a maximum matching, an algorithm due to Dinitz [Din70] finds a maximal set of edge-disjoint augmenting paths by conducting a depth-first search of the layer graph $\mathcal{L}$. Efrat and Itai's algorithm takes advantage of the geometric setting of the graph to improve the efficiency of Dinitz's algorithm. For this an abstract data structure $D_\epsilon(S)$ is used, for a set of points S. Using this data structure the set of vertices of each layer L_i is found, without explicitly constructing all the edges of $\mathcal{L}$. Given a set of points S and a real number $\epsilon > 0$, the data structure should support two operations :

(i) neighbor$_\epsilon(S, q)$: For query point q, return a point $s \in S$ whose distance from q is atmost ϵ. If no such s exists, then neighbor$_\epsilon(S, q) = \emptyset$.

(ii) delete$_\epsilon(S, s)$: Delete the point s from S.

Let $T(|S|)$ denote an upper bound on the time of performing one of these operations. At this moment let us disregard the time required to construct this data structure, since it will be shown in the next subsection that this time in our problem is bounded by $O(nT(n))$, where S is a point set in 3-D with cardinality n. Hence it will not influence the overall time complexity. Using this data structure, Algorithm 4.1 shows how the layer graph $\mathcal{L}$ is constructed. It is to be noted that each matched vertex of A is reached in constant time from its pair in M, and each vertex of B is found atmost once by a query of neighbor$_\epsilon(D, \cdot)$ and also deleted from D atmost once. Hence constructing $\mathcal{L}$ requires $O(nT(n))$ time.

Algorithm 4.1 (Efrat and Itai)

Input : *Graph G arising out of point sets A, B, and real number $\epsilon > 0$, and a matching M.*

Output : *Layer graph $\mathcal{L}$*

L_1 = exposed vertices of A;

$i = 1$;

```
D = D_ε(B);
repeat forever
{
  L_2i = ∅;
  for each a ∈ L_2i−1
    while neighbor_ε(D, a) ≠ ∅
    {
      b = neighbor_ε(D, a);
      add b to L_2i;
      delete_ε(D, b);
    }
  if L_2i is empty then { no augmenting path exists; Stop; }
  else
  {
    if L_2i contains exposed vertices
    then { construction of L is complete; return L; }
    else L_2i+1 = all vertices of A adjacent to L_2i via edges of M;
  }
  i = i + 1;
}
```

Next we describe the procedure for finding augmenting paths from the exposed vertices of L_1 to exposed vertices of L_{2t}. For this we need the following lemma.

Lemma 4.1 (Efrat and Itai) *Let M be a graph-matching of a bipartite graph $G = (A \cup B, E)$, let Π be a set of edge-disjoint augmenting paths, and v be an intermediate vertex of some path of Π. Then v cannot participate in any other augmenting path of Π.*

To look for augmenting paths, $D_{2i} \equiv D_\epsilon(L_{2i})$ is first constructed for each of the even layers $L_{2i} \subseteq B$ of the layer graph $\mathcal{L}$. Then a depth-first search is conducted, starting from an exposed vertex of L_1. To advance from a vertex $a \in L_{2i-1}$, the

operation neighbor$_\epsilon(D_{2i}, a)$ is invoked on the data structure D_{2i}. If this returns a vertex $b \in L_{2i}$ then (a, b) is added to the current path and we advance to b. If neighbor$_\epsilon(D_{2i}, a)$ returns $\emptyset$, it indicates that a does not have any neighbors in L_{2i} and hence does not lead to an exposed vertex of L_{2t}, so we should backtrack. To advance from $b \in L_{2i}$ $(i < t)$, if $(b, a^+) \in M$ then (b, a^+) is added to the path and we advance from a^+. Note that b is never exposed since all exposed vertices of $B \cap \mathcal{L}$ belong to L_{2t}. If $b \in L_{2t}$ is an exposed vertex, then an augmenting path is found. In this case M is increased and the vertices of this augmenting path from the appropriate L_is are deleted (because of Lemma 4.1), before starting to look for another augmenting path. To backtrack from $a \in L_{2i-1}$ $(i \geq 2)$, if $(b^-, a) \in M$ and a^- be the vertex preceeding b^- on the path found so far, then a and b^- are removed from the path and the search is continued from a^-. If $a \in L_1$ then it is simply deleted from L_1.

When no exposed vertices remain in L_1, the search for augmenting paths in this particular phase is complete and we proceed to the next phase, where the layer graph $\mathcal{L}$ is again constructed and the whole process repeated. If during the construction of $\mathcal{L}$ one does not reach any exposed vertices of B, then it indicates that a maximum matching has been found.

The time required to find all alternating paths in a single layer graph is $O(nT(n))$. The time required to construct a layer graph was also shown to be $O(nT(n))$. A theorem due to Hopcroft and Karp [HK73] states that Dinitz's matching algorithm requires $O(\sqrt{n})$ phases. Hence Efrat and Itai's scheme requires $O(n^{1.5}T(n))$ time.

4.1.2 Faster Algorithms for Common Point Set Identification

Making use of the scheme just described, in this subsection we show that replacing the Hopcroft and Karp's algorithm in our approximation algorithms for common point set identification leads to an improvement in the overall running time. Recall that Algorithm 3.1 for deciding if there exists an α-$LCP(A, B, \epsilon)$, had an indecision interval of $[\frac{1}{8}\epsilon_{min}(\alpha), 8\epsilon_{min}(\alpha))$ and time complexity $O(n^{8.5})$ if A and B are point sets of cardinality $O(n)$. We shall modify this algorithm to illustrate the improvement in

running time by using our new scheme. In that process it will also become clear that the same modification is extendable to all the other algorithms in a straightforward manner, to obtain the same speedup.

First we describe the data structure $D_\epsilon(S)$ for our problem, that is made use of in Efrat and Itai's algorithm. Arya *et al.* [AMN+94] developed a data structure for answering nearest neighbor queries for a set of points in $\Re^d$. Given a set of n points S in $\Re^d$, and given any point $q \in \Re^d$, a nearest neighbor to q in S is any point $s \in S$ that minimizes $d(s,q)$. For $d \geq 3$, there is no known algorithm for answering nearest neighbor queries that achieves both nearly linear space and polylogarithmic query time. The data structure proposed by Arya *et al.* reports in time $O(\log n)$, an approximated nearest neighbor of a query point $q \in \Re^d$. Approximate nearest neighbor is defined as a point $s \in S$, such that for all $s' \in S$, $d(s,q) \leq (1+\delta)d(s',q)$, where $\delta > 0$ is a predefined parameter. The construction of this data structure takes $O(n \log n)$ time, $O(n)$ space, and can also be dynamized so that deletion takes $O(\log n)$ time.

Our new approximation algorithm for common point set identification uses the Efrat and Itai's algorithm with Arya *et al.*'s data structure for finding the maximum matching, instead of the Hopcroft and Karp's algorithm as was done in the last chapter. Thus $T(n)$, the upper bound on performing the operations $\text{neighbor}_\epsilon(S,q)$ and $\text{delete}_\epsilon(S,s)$ is $O(\log n)$. Recall from the last subsection that Efrat and Itai's algorithm requires time $O(n^{1.5}T(n))$. Hence the new, but approximate matching algorithm has time complexity $O(n^{1.5}\log n)$ instead of $O(n^{2.5})$. Let us refer to the new graph matching algorithm as $AM()$ (Approximate Matching), which takes as input two point sets A, B, and real numbers $\epsilon > 0$ and $\delta > 0$. To implement $\text{neighbor}_\epsilon(S,q)$, Arya *et al.*'s data structure is used to find a point s. If $d(s,q) \leq (1+\delta)\epsilon$, then s is reported as $\text{neighbor}_\epsilon(S,q)$, otherwise $\text{neighbor}_\epsilon(S,q)$ is $\emptyset$. We will show that the consequence of using this approximate matching in our decision algorithms is an increase in the indecision interval, along with a one sided error for a small range of values of the point location error ϵ, both of which depend on δ. This drawback is also reflected in the results obtained, by modifying this decision algorithm to approximate the size of $\alpha_{max}(\epsilon)$, as we did in the last chapter. Recalling

the definitions of T_{PQ} from Definition 3.3, and $G(T_{PQ}, \epsilon, A, B)$ from Definition 3.4, for triplets of points P and Q, we now state our algorithm and the consequent theorem.

Algorithm 4.2

Input : *Point sets A, B, real numbers $\epsilon > 0$, $0 < \alpha \leq 1$, $\delta > 0$.*

Output : *If α-$LCP(A, B, \epsilon)$ exists then YES or DON'T KNOW, else NO or DON'T KNOW.*

for all triplets $P = (p_{i_1}, p_{i_2}, p_{i_3})$ from A
  for all triplets $Q = (q_{j_1}, q_{j_2}, q_{j_3})$ from B
  {
    compute T_{PQ};
    $M = AM(T_{PQ}(A), B, \epsilon, \delta)$;
    if $(M \geq \alpha \min(|A|, |B|))$ **then return** YES;
  }
decision = NO;
for all triplets $P = (p_{i_1}, p_{i_2}, p_{i_3})$ from A
  for all triplets $Q = (q_{j_1}, q_{j_2}, q_{j_3})$ from B
  {
    compute T_{PQ};
    $M = AM(T_{PQ}(A), B, 8\epsilon, \delta)$;
    if $(M \geq \alpha \min(|A|, |B|))$ **then** decision = YES;
  }
if (decision == NO) **then return** NO **else return** DON'T KNOW;

Theorem 4.2 *Algorithm 4.2 always returns the correct answer about the existance of α-$LCP(A, B, \epsilon)$ if*

(i) $\epsilon \geq 8\epsilon_{min}(\alpha)$, **or**

(ii) $\epsilon < \frac{\epsilon_{min}(\alpha)}{8(1+\delta)}$

It either returns the correct answer or returns DON'T KNOW for values of $\epsilon \in [\frac{\epsilon_{min}(\alpha)}{8(1+\delta)}, \frac{\epsilon_{min}(\alpha)}{1+\delta}) \cup [\epsilon_{min}(\alpha), 8\epsilon_{min}(\alpha))$, *and for* $\epsilon \in [\frac{\epsilon_{min}(\alpha)}{(1+\delta)}, \epsilon_{min}(\alpha))$ *it might return*

any of the three possible answers - YES, NO, DON'T KNOW. A transformatio T_{PQ}, along with the bijective mapping l induced by the matching algorithm $AM(\,)$ that results in Algorithm 4.2 to return YES, correspond to α-$LCP(A, B, (1+\delta)\epsilon)$.

Proof: It follows from Lemma 3.1 that Algorithm 4.2 finds a transformation T_{PQ} such that atleast $\alpha \min(|A|, |B|)$ points of $T_{PQ}(A)$ are within $8\epsilon_{min}(\alpha)$ distance of distinct points of B. So the graph $G(T_{PQ}, \epsilon, A, B)$ always has a matching of size greater than or equal to $\alpha \min(|A|, |B|)$ if $\epsilon \geq 8\epsilon_{min}(\alpha)$. However, the approximate matching algorithm $AM(\,)$ considers a graph $G \supseteq G(T_{PQ}, \epsilon, A, B)$ instead of $G(T_{PQ}, \epsilon, A, B)$. This is because some edges of G are larger than ϵ, but none are longer than $(1+\delta)\epsilon$ as apparent from the discussion in the last subsection. $AM(\,)$ thus finds a maximum matching in G where $G(T_{PQ}, \epsilon, A, B) \subseteq G \subseteq G(T_{PQ}, (1+\delta)\epsilon, A, B)$. Any matching in $G(T_{PQ}, \epsilon, A, B)$ is also a matching in G, and if a matching in $G(T_{PQ}, \epsilon, A, B)$ can be increased by an augmenting path then for a matching of the same size there also exists an augmenting path in G. Thus if $G(T_{PQ}, \epsilon, A, B)$ has a matching of size k, then the approximate matching algorithm $AM(\,)$ finds a matching of size $\geq k$. Hence the algorithm returns YES for any $\epsilon \geq 8\epsilon_{min}(\alpha)$. But since a maximum matching by $AM(\,)$ might have some edges of length between ϵ and $(1+\delta)\epsilon$, the labelling induced by $AM(\,)$ along with T_{PQ} correspond to α-$LCP(A, B, (1+\delta)\epsilon)$.

For any $\epsilon < \frac{1}{8}\epsilon_{min}(\alpha)$, no isometry enables in finding α-$LCP(A, B, 8\epsilon)$. But since $AM(\,)$ considers the graph $G \subseteq G(T_{PQ}, 8(1+\delta)\epsilon, A, B)$, the indecision interval extends to $\frac{\epsilon_{min}(\alpha)}{8(1+\delta)}$ on the left. If $G(T_{PQ}, 8(1+\delta)\epsilon, A, B)$ does not contain a matching of size k, then G also does not have a matching of size k. For any $\epsilon < \frac{\epsilon_{min}(\alpha)}{8(1+\delta)}$ there does not exist any isometry T_{PQ} such that $G(T_{PQ}, 8(1+\delta)\epsilon, A, B)$ has a matching of size $\alpha \min(|A|, |B|)$. Hence for such ϵ Algorithm 4.2 always returns NO.

For values of $\epsilon \in [\frac{\epsilon_{min}(\alpha)}{1+\delta}, \epsilon_{min}(\alpha))$, no isometry T_{PQ} enables in finding α-$LCP(A, B, \epsilon)$. But since $AM(\,)$ considers a graph $G \subseteq G(T_{PQ}, (1+\delta)\epsilon, A, B)$, G might have a matching of size $\geq \alpha \min(|A|, |B|)$. Hence the algorithm might return YES. The arguments for it returning NO or DON'T KNOW are exactly similar to those given for Theorem 3.2.

For any $\epsilon \in [\frac{\epsilon_{min}(\alpha)}{8(1+\delta)}, \frac{\epsilon_{min}(\alpha)}{1+\delta})$, there does not exist any transformation T_{PQ} for which the graph $G(T_{PQ}, (1+\delta)\epsilon, A, B)$ has a matching of size $\geq \alpha \min(|A|, |B|)$.

Since $AM()$ considers a graph $G \subseteq G(T_{PQ}, (1+\delta)\epsilon, A, B)$, it can never return a matching of this size. Hence for such ϵ the algorithm never returns YES. For $\epsilon \in [\epsilon_{min}(\alpha), 8\epsilon_{min}(\alpha))$, it follows from Lemma 3.1 that the algorithm finds a transformation T_{PQ}, for which $G(T_{PQ}, 8\epsilon, A, B)$ has a matching of size $\geq \alpha \min(|A|, |B|)$. For such a transformation $AM()$ also finds a matching of atleast this size. Hence the algorithm never returns NO for such values of ϵ. ∎

Theorem 4.3 *If A and B are point sets of cardinality $O(n)$ then Algorithm 4.2 runs in $O(n^{7.5} \log n)$ time.*

The corresponding algorithm that we gave in the last chapter, i.e. Algorithm 3.1, had a running time of $O(n^{8.5})$. However, as spelled out by Theorem 4.2, the new algorithm suffers from the following shortcomings : (i) the indecision interval increases from $[\frac{1}{8}\epsilon_{min}(\alpha), 8\epsilon_{min}(\alpha))$ to $[\frac{\epsilon_{min}(\alpha)}{8(1+\delta)}, \frac{\epsilon_{min}(\alpha)}{1+\delta}) \cup [\epsilon_{min}(\alpha), 8\epsilon_{min}(\alpha))$ (ii) although α-$LCP(A, B, \epsilon)$ does not exist for any $\epsilon \in [\frac{\epsilon_{min}(\alpha)}{1+\delta}, \epsilon_{min}(\alpha))$ Algorithm 4.2 might return YES for such values of ϵ. However, the corresponding Algorithm 3.1 never returns a wrong answer (iii) for any specified point location error ϵ, if the algorithm returns YES then corresponding transformation along with the bijective mapping induced by the graph matching algorithm $AM()$ correspond to α-$LCP(A, B, (1+\delta)\epsilon)$. There was no such increase in point location error in the case of Algorithm 3.1.

In Theorem 3.5 we saw that while trying to approximate $\alpha_{max}(\epsilon)$, the lower bound α_l obtained by Algorithm 3.2 was $\leq \alpha_{max}(\epsilon)$ and $\alpha_l \geq \max\{\alpha : \epsilon \geq 8\epsilon_{min}(\alpha)\}$ (recall that if $\alpha_1 \leq \alpha_2$ then $\epsilon_{min}(\alpha_1) \leq \epsilon_{min}(\alpha_2)$). However, if we modify Algorithm 4.2 in the same way as was done for Algorithm 3.1, the lower bound α_l might be greater than $\alpha_{max}(\epsilon)$. But since the graph considered by $AM()$ can have a matching of size atmost $\alpha_{max}((1+\delta)\epsilon)\min(|A|, |B|)$, α_l will be $\leq \alpha_{max}((1+\delta)\epsilon)$. Hence we have $\max\{\alpha : \epsilon > 8\epsilon_{min}(\alpha)\} \leq \alpha_l \leq \alpha_{max}((1+\delta)\epsilon)$. Using the same reasoning given in Theorem 4.2, $\alpha_{max}(\epsilon) < \alpha_u \leq \min\{\alpha : \epsilon < \frac{\epsilon_{min}(\alpha)}{8(1+\delta)}\}$.

To reduce the indecision interval of $[\frac{1}{8}\epsilon_{min}(\alpha), 8\epsilon_{min}(\alpha))$ of Algorithm 3.1 to $[\frac{1}{2}\epsilon_{min}(\alpha), 2\epsilon_{min}(\alpha))$ in Algorithm 3.4, we used our exact algorithm for finding the largest common point set under pure rotation i.e. Algorithm 3.3. All the remaining algorithms of Chapter 3 hinge on this algorithm. Note that the graph matching algorithm is invoked within this exact algorithm. Replacing the Hopcroft and

Karp's algorithm by the approximate matching algorithm $AM()$ will reduce the running time of this algorithm from $O(n^{6.5})$ to $O(n^{5.5}\log n)$, and thereby speedup the overall running time of all the algorithms of the last chapter which make use of it. Recall from the last chapter that in this algorithm, given point sets A, B, a fixed point p, and a specified point location ϵ, we constructed all possible domes D_{ij} for pairs of points $a_i \in A$ and $b_j \in B$. These domes partitioned the surface of the sphere S_p around p, into a number of regions. Each point within a region corresponds to a rotation R such that if this region is formed by the intersection of the domes $D_{i_1j_1}, D_{i_2j_2}, \ldots, D_{i_kj_k}$, then $d(a_{i_l}, b_{j_l}) \leq \epsilon$ for $a_{i_l} \in R(A)$, $b_{j_l} \in B$, $l = 1, 2, \ldots, k$. This gives rise to a bipartite graph where the nodes of the graph are points of A and B and the edges join pairs (a_{i_l}, b_{j_l}), $l = 1, 2, \ldots, k$. Clearly, every rotation within a region gives rise to the same graph. To apply our approximate matching algorithm $AM()$ to find a maximum matching in such a graph, we require a rotation corresponding to any point in the region that gave rise to this graph. Note that in our sweep algorithm, that was used to traverse all the regions on the surface of S_p, we constructed the bipartite graph corresponding to event points which were either intersection points, or the leftmost points of the domes. To replace the Hopcroft and Karp's algorithm, we explicitly compute the rotation corresponding to each such event point, apply this rotation on the set A, and then use our approximate matching algorithm $AM()$. If R be the rotation found by Algorithm 3.3 that enables in finding $\alpha_{max}(\epsilon)$-$LCP(A, B, \epsilon)$, under pure rotation, it implies that the graph $G(R, \epsilon, A, B)$ has a maximum matching of size $\alpha_{max}(\epsilon)\min(|A|, |B|)$, and it was found out by the Hopcroft and Karp's algorithm. However, since our approximate graph matching algorithm $AM()$ considers a graph G, where $G(R, \epsilon, A, B) \subseteq G \subseteq G(R, (1+\delta)\epsilon, A, B,)$, it might find a matching of size larger than the maximum matching in $G(R, \epsilon, A, B)$. But for any rotation R, $G(R, (1+\delta)\epsilon, A, B)$ can have a matching of size atmost $\alpha_{max}((1+\delta)\epsilon)\min(|A|, |B|)$. So our new algorithm for finding the largest common point set under rotation returns a value α, where $\alpha_{max}(\epsilon) \leq \alpha \leq \alpha_{max}((1+\delta)\epsilon)$. Use of this algorithm in Algorithm 3.4, has consequences similar to those seen in the case of Algorithm 4.2 in the last two paragraphs. Recall that Algorithm 3.4 had an indecision interval of

$[\frac{1}{2}\epsilon_{min}(\alpha), 2\epsilon_{min}(\alpha))$ and time complexity of $O(n^{8.5})$. Using our new algorithm for finding the largest common point set under rotation, changes the indecision interval to $[\frac{\epsilon_{min}(\alpha)}{2(1+\delta)}, \frac{\epsilon_{min}(\alpha)}{1+\delta}) \cup [\epsilon_{min}(\alpha), 2\epsilon_{min}(\alpha))$. Moreover the algorithm might err for values of ϵ belonging to $[\frac{\epsilon_{min}(\alpha)}{1+\delta}, \epsilon_{min}(\alpha))$. However, this new algorithm runs in time $O(n^{7.5}\log n)$. Using the same reasoning followed in the last two paragraphs, if this algorithm is made use of to approximate $\alpha_{max}(\epsilon)$, the lower and upper bounds α_l and α_u will be as follows : $\max\{\alpha : \epsilon > 2\epsilon_{min}(\alpha)\} \leq \alpha_l \leq \alpha_{max}((1+\delta)\epsilon)$ and $\alpha_{max}(\epsilon) < \alpha_u \leq \min\{\alpha : \epsilon < \frac{\epsilon_{min}(\alpha)}{2(1+\delta)}\}$. Exactly similar results are obtained in the case of Algorithms 3.5 and 3.6 of the last chapter.

4.2 Using Random Sampling

In this section we show that application of standard random sampling techniques can reduce the time complexity of our algorithms by a considerable extent, however, at the cost of a small failure probability. By now this technique has become fairly standard for this class of problems [ATT97, FKL+97, IR96]. We had briefly mentioned in Chapter 2 how this is done in the case of the alignment scheme, which is the basis of all our algorithms, except our exact algorithm for finding the largest common point set under rotation, i.e. Algorithm 3.3. The basic idea is that while searching for a transformation that enables in finding the largest common point set between two given point sets, instead of going through all possible transformations, only those arising out of a randomly sampled set of points are explored. If the size of the common point set is a considerable fraction of the size of the given point sets, then with a high probability we pick up points lying within the common point set, which thus enables in finding out a required transformation. Clearly, this technique can be directly applied to Algorithm 3.1, .where instead of going through all possible triplets of points, only those corresponding to randomly sampled subsets of A and B are tested. However, recall from Theorem 3.2 that it was possible to bound the indecision interval of Algorithm 3.1, due to a transformation T_{PQ} where P is a triplet that 'enclosed' the points of A belonging to the common point set. Since the algorithm went through all possible triplets of A, such a triplet was guaranteed

to be considered. We could not prove such a bound on the indecision interval in the randomized case. However for the remaining algorithms of Chapter 3, the approximation ratios do hold in their corresponding randomized counterparts which we state below. We also bound the probability of failure in all the cases.

Recall that all the algorithms of the last chapter except the first three, in the process of searching for a transformation that enables in finding the largest common point set, seperately computed the translation and the rotation components of the transformation. Towards this, for pairs of points $a \in A$ and $b \in B$, if t_{ab} denotes that translation that takes a to b, then all possible translations t_{ab} arising out of the point sets A and B were explored. For each such translation, our exact algorithm for finding out the largest common point set under rotation was invoked. Our randomized algorithms are based on the scheme of exploring all translations corresponding to randomly sampled subsets of the given point sets, instead of the original ones. If the size of these randomly sampled subsets is smaller than that of the original sets, then this leads to fewer invocations of the algorithm for computing the rotation and hence reduces the overall running time. The speedup obtained depends on the ratio of the size of the original sets to that of the the sampled subsets. By carefully choosing the size of the subsets, we can bound the probability of failure of such a scheme. We illustrate this scheme by modifying Algorithm 3.4, and show how this affects the overall running time and failure probability. As we proceed, it will become clear that the same results will also hold in case of all the other algorithms.

Our first use of randomization is in choosing a subset of the given point set A. Towards this we have the following algorithm and the consequent theorem.

Algorithm 4.3
Input : *Point sets A, B, real numbers $\epsilon > 0$, $0 < \alpha \leq 1$.*
Output : *Either YES or NO or DON'T KNOW.*

A' = a multiset of cardinality k, containing randomly sampled points of A;
for each point $a' \in A'$
  for each point $b \in B$
  {

```
    α' = Algorithm 3.3(t_{a'b}(A), B, b, ε);
    if (α' ≥ α) then return YES;
  }
decision = NO;
for each point a' ∈ A'
  for each point b ∈ B
  {
    α' = Algorithm 3.3(t_{a'b}(A), B, b, 2ε);
    if (α' ≥ α) then decision = YES;
  }
if (decision = = NO) then return NO else return DON'T KNOW;
```

Before stating the theorem, let us assume that A and B are point sets of equal cardinality n. In the case of A and B being of unequal cardinality, the same type of analysis that we present here will follow. But this assumption leads to simplified expressions.

Theorem 4.4 *If point sets A and B are of cardinality n, and the cardinality of the randomly sampled multiset A' be a constant k, then Algorithm 4.3 runs in time $O(n^{7.5})$. For any $k \geq \lceil \frac{1}{\alpha} \ln \frac{1}{1-q} \rceil$, the algorithm returns YES with probability atleast q, for all $\epsilon \geq 2\epsilon_{min}(\alpha)$. For $\epsilon < \frac{1}{2}\epsilon_{min}(\alpha)$ the algorithm always returns NO, and for $\frac{1}{2}\epsilon_{min}(\alpha) \leq \epsilon < \epsilon_{min}(\alpha)$ it either returns NO or DON'T KNOW.*

Proof: Since Algorithm 3.3 has a time complexity of $O(n^{6.5})$ it follows that Algorithm 4.3 runs in time $O(n^{7.5})$. If $|A' \cap \alpha\text{-}LCP(A, B, \epsilon_{min}(\alpha))| \geq 1$, then it follows from Theorem 3.8 that Algorithm 4.3 returns YES for all $\epsilon \geq 2\epsilon_{min}(\alpha)$. We show that for $\Pr(\ |A' \cap \alpha\text{-}LCP(A, B, \epsilon_{min}(\alpha))| \geq 1\ ) \geq q$ to hold, it is sufficient that $k \geq \frac{1}{\alpha} \ln \frac{1}{1-q}$.

Let a be a randomly sampled element of A. Then $\Pr(\ a \in \alpha\text{-}LCP(A, B, \epsilon_{min}(\alpha))\ ) \geq \alpha$. Since A' is of cardinality k, $\Pr(\ A' \cap \alpha\text{-}LCP(A, B, \epsilon_{min}(\alpha) = \emptyset)\ ) < (1-\alpha)^k$. Hence for $\Pr(\ |A' \cap \alpha\text{-}LCP(A, B, \epsilon_{min}(\alpha))| \geq 1\ ) \geq q$ to hold, it is sufficient that,

$$1 - (1-\alpha)^k \geq q$$

Rearranging and taking logarithms of both sides,

$$k \ln(1-\alpha) \leq \ln(1-q)$$

Expanding the logarithmic term,

$$k(\alpha + \frac{\alpha^2}{2} + \frac{\alpha^3}{3} + \ldots) \geq \ln \frac{1}{1-q}$$

For this inequality to hold, it is sufficient that

$$k \geq \frac{1}{\alpha} \ln \frac{1}{1-q}$$

For any $\epsilon < \epsilon_{min}(\alpha)$, no isometry can result in finding α-$LCP(A, B, \epsilon)$. Therefore for such ϵ the algorithm never returns YES. Moreover, for $\epsilon < \frac{1}{2}\epsilon_{min}(\alpha)$ no isometry enables in finding α-$LCP(A, B, 2\epsilon)$. Thus such values of ϵ always result in Algorithm 4.3 to return NO. ∎

Our second randomized algorithm extends this concept of testing translations corresponding to only a randomly sampled subset of A. In this case we not only randomly sample a subset of A, but also randomly sample a subset of B and then test only the translations corresponding to these subsets. Rest of the algorithm is exactly similar to Algorithm 4.3. If A' and B' are randomly sampled multisets of A and B, then we invoke Algorithm 3.3 for all possible translations $t_{a'b'}$, $a' \in A'$ and $b' \in B'$. We thus have the following theorem.

Theorem 4.5 *If point sets A and B are of cardinality n, and the cardinality of the randomly sampled multisets A' and B' be constants k_1 and k_2 respectively, then the algorithm runs in time $O(n^{6.5})$. For $k_1 k_2 \geq$ any arbitrary constant c, the algorithm returns YES with probability atleast $1 - e^{-\frac{c\alpha}{n}}$, for all $\epsilon \geq 2\epsilon_{min}(\alpha)$. For $\epsilon < \frac{1}{2}\epsilon_{min}(\alpha)$ the algorithm always returns NO, and for $\frac{1}{2}\epsilon_{min}(\alpha) \leq \epsilon < \epsilon_{min}(\alpha)$ it either returns NO or DON'T KNOW.*

Proof: Let l be the bijective mapping underlying α-$LCP(A, B, \epsilon_{min}(\alpha))$. It follows from Theorem 3.8 that for any $\epsilon \geq 2\epsilon_{min}(\alpha)$, the algorithm always returns YES if $A' \cap \alpha$-$LCP(A, B, \epsilon_{min}(\alpha)) \neq \emptyset$ and $l(a) \in B'$, where a is any element of

$A' \cap \alpha\text{-}LCP(A, B, \epsilon_{min}(\alpha))$. We show that for both the conditions to be satisfied with probability atleast $1 - e^{-\frac{c\alpha}{n}}$, it is sufficient that $k_1 k_2 \geq c$. The rest of the proof is similar to that for Theorem 4.4.

Let a be a randomly sampled element of A. Then $\Pr(\, a \in \alpha\text{-}LCP(A, B, \epsilon_{min}(\alpha))\,) \geq \alpha$. Assuming that $a \in \alpha\text{-}LCP(A, B, \epsilon_{min}(\alpha))$, let E_1 denote the event that the multiset B' contains $l(a)$. Then, $\Pr(E_1) = 1 - (\frac{n-1}{n})^{k_2}$. Let E_2 denote the event, that in the process of picking up a random element $a \in A$ and then randomly picking up k_2 elements of B with replacement, we never come across a pair $a \in A$, $b \in B$ such that $a \in \alpha\text{-}LCP(A, B, \epsilon_{min}(\alpha))$ and $b = l(a)$. Then $\Pr(E_2) < 1 - \alpha\left\{1 - \left(\frac{n-1}{n}\right)^{k_2}\right\}$. Let E_3 denote the event that in the process of randomly sampling k_1 elements of A with replacement, and for each such element randomly sampling k_2 elements of B with replacement, we come across atleast one pair $a \in A$, $b \in B$, such that $a \in \alpha\text{-}LCP(A, B, \epsilon_{min}(\alpha))$ and $l(a) = b$. This is precisely the event that we are looking for. Clearly, $\Pr(E_3) \geq 1 - \left[1 - \alpha\left\{1 - (\frac{n-1}{n})^{k_2}\right\}\right]^{k_1}$. We want this to be atleast $1 - e^{-\frac{c\alpha}{n}}$ i.e.

$$\left[1 - \alpha\left\{1 - \left(\frac{n-1}{n}\right)^{k_2}\right\}\right]^{k_1} \leq e^{-\frac{c\alpha}{n}}$$

This is equivalent to,

$$k_1 \ln\left(1 - \alpha\left\{1 - \left(\frac{n-1}{n}\right)^{k_2}\right\}\right) \leq -\frac{c\alpha}{n}$$

Expanding the ln term gives

$$k_1\alpha\left\{1 - \left(\frac{n-1}{n}\right)^{k_2}\right\}\left[1 + \frac{\alpha}{2}\left\{1 - \left(\frac{n-1}{n}\right)^{k_2}\right\} + \ldots\right] \geq \frac{c\alpha}{n}$$

For this to hold it is sufficient that

$$k_1\alpha\left\{1 - \left(\frac{n-1}{n}\right)^{k_2}\right\} \geq \frac{c\alpha}{n}$$

Expanding $(\frac{n-1}{n})^{k_2}$ gives

$$k_1\left(\frac{k_2}{n} - \frac{k_2(k_2-1)}{2n^2} + \ldots\right) \geq \frac{c}{n}$$

Hence, $k_1 k_2 \geq c$. ∎

4.3 Practical Algorithms Exploiting the Stuctural Properties of Proteins

All the algorithms presented so far have time complexity which are relatively high degree polynomials of the size of the point sets. Recall from the last chapter that these 3-D point sets actually represent some drug or protein molecules, where each point is representative of an atom of the molecule. Since drug molecules are quite small in size, ranging from ten to a few hundred atoms, in practice these algorithms will be quite fast. However, since protein molecules are usually of the order of a few thousand atoms, in many practical situations such as database queries, the running time of our algorithms might be unacceptable. Till now we have not used the fact that our point sets are actually representative of some molecules. Since each atom of a molecule can be mapped only to a similar atom of the other molecule, this reduces the total number of possible transformations by a large extent. However, it is difficult to make use of this fact in the analysis of the running time of our algorithms in a general setting. In the case of particular molecules, by using a quantitative information about their structure, it will be possible to bound the exact running times apriori. In this section we show that by exploiting some general structural properties of proteins, it is possible to design algorithms with much smaller time complexities compared to those presented so far, however, at the cost of loosing the guaranteed performance bounds. But we expect that for all practical purpose they would perform quite well. Our algorithms are completely general and can be used for the common substructure identification in case of any protein molecules.

4.3.1 Protein Structure

A protein is composed of a chain of amino acids linked to each other by peptide bonds. There are three groups of atoms in each amino acid which constitute the backbone of the chain : the central atom C_α, to which are attached on each side an N-H group and a carbonyl group $C' = O$. The alkyl residue R, also bound to the C_α, characterizes the nature of the amino acid but does not take part in the backbone of the chain (see Figure 6). We will refer to the N-H group by the N

atom and the $C' = O$ group by the C' atom.

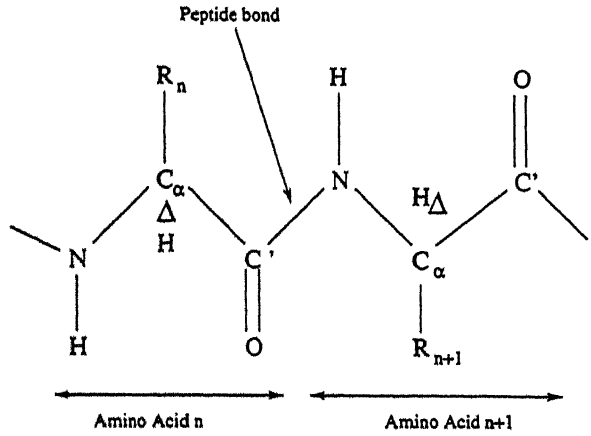


Figure 6: Structure of a Protein Chain

The sequence of amino acids fold in space to generate a complex three dimensional structure. Although there can be rotations around the C_α-C' and C_α-N bonds, and hence the geometry of the chain is weakly constrained, the geometry of the atoms attached to C_α is perfectly determined. The three atoms N, C_α and C' in each amino acid form a triangle which uniquely defines the position and orientation of the amino acid, and hence the entire protein structure, in space. Thus all the N, C_α and C' atoms act as a backbone or skeleton to which the alkyl residues R are attached. Since the C_α-N and C_α-C' bond lengths and the $NC_\alpha C'$ bond angle are fixed, the skeletons corresponding to two common substructures of two proteins will be exactly congruent. Since correspondence between two triplets of points is sufficient to uniquely determine a rigid transformation in space, if we know the correspondence between two amino acids belonging to the common substructures of two protein molecules then we can compute the rigid transformation that results in the two skeletons to exactly coincide. We make use of this property in our algorithm in the next subsection.

4.3.2 An $O(n^{4.5})$ Algorithm Using Protein Structure Information

Given point sets A, B, and a real number $\epsilon > 0$, let l be the bijective mapping underlying $\alpha_{max}(\epsilon)$-$LCP(A, B, \epsilon)$. Let $p_1 \in \alpha_{max}(\epsilon)$-$LCP(A, B, \epsilon)$, p_2 be the point of $\alpha_{max}(\epsilon)$-$LCP(A, B, \epsilon)$ which is farthest from p_1, and p_3 be the point of $\alpha_{max}(\epsilon)$-$LCP(A, B, \epsilon)$ such that the perpendicular distance of p_3 from the line $\overline{p_1p_2}$ is maximized. Now consider the isometric transformation $\mathcal{I}$ such that $\mathcal{I}(p_1) = l(p_1)$; $\mathcal{I}(p_1)$, $\mathcal{I}(p_2)$ and $l(p_2)$ are collinear; $\mathcal{I}(p_1)$, $\mathcal{I}(p_2)$, $\mathcal{I}(p_3)$ and $l(p_3)$ are coplanar. Recall from Lemma 3.1 that the isometry $\mathcal{I}$ and the bijective mapping l correspond to $\alpha_{max}(\epsilon)$-$LCP(A, B, 8\epsilon)$. Note that the triplet (p_1, p_2, p_3) defines a cylindrical region $\{p \in \Re^3 : |\overline{p_1p}| \leq |\overline{p_1p_2}|$ and $\text{dist}(p, \overline{p_1p_2}) \leq \text{dist}(p_3, \overline{p_1p_2})\}$ (see Figure 7), where $\text{dist}(p, \overline{p_1p_2})$ denotes the perpendicular distance of the point p from the straight line through p_1 and p_2.

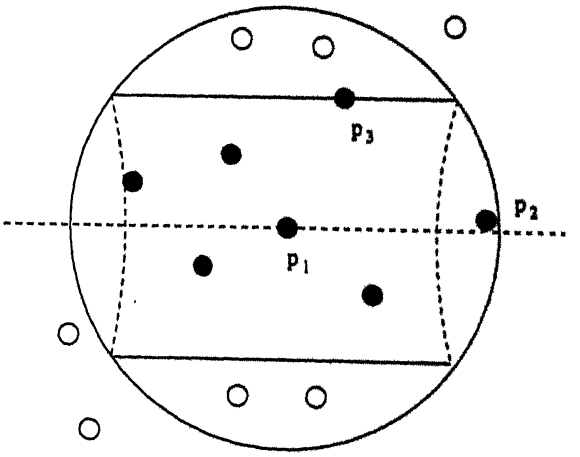


Figure 7: The cylindrical region defined by the points p_1, p_2 and p_3

All points of $\alpha_{max}(\epsilon)$-$LCP(A, B, \epsilon)$ lie on or within this cylindrical region and are atmost 8ϵ distance away from their corresponding points in the set B, under the isometry $\mathcal{I}$. In the case of our protein molecules, let p_1, p_2, p_3 be either N, C_α, or C' atoms lying on the skeleton or backbone of the common substructure and define the cylindrical region which enclose the entire skeleton of the substructure. Then the only parts of $\alpha_{max}(\epsilon)$-$LCP(A, B, \epsilon)$ that can be outside this cylindrical region are some of the alkyl residues bound to the C_α atoms lying on or within

this cylindrical region. Hence the isometry $\mathcal{I}$ in this case takes each point of the common substructure to within 8ϵ distance of its corresponding point in the set B, except possibly for some of the points corresponding to the alkyl residues which lie outside this cylindrical region. As mentioned in the last subsection, since the skeletons corresponding to the common substructures of two proteins are exactly congruent, any isometric transformation which maps three points on this skeleton to the corresponding points of the set B, will satisfy the properties of isometry $\mathcal{I}$. Hence we have the following algorithm.

Algorithm 4.4

Input : *Labelled point sets A and B corresponding to two protein molecules and a real number $\epsilon > 0$.*

Output : *A real number α.*

$M = 0$;
for each amino acid $a \in A$
  for each amino acid $b \in B$
  {
    T_{ab} = transformation that takes the N, C_α and C' atoms of a to the corresponding atoms of b;
    M' = size of the maximum matching in $G(T_{ab}, 8\epsilon, A, B)$;
    if $(M' > M)$ **then** $M = M'$;
  }
  return $M/\min(|A|, |B|)$;

Clearly, if the algorithm returns α, then there is a subset S of A of cardinality $\alpha \min(|A|, |B|)$ which is 8ϵ-congruent to some subset of B. If the skeleton or backbone corresponding to the points of S, formed by the N, C_α and C' atoms, enclose all the points of $\alpha_{max}(\epsilon)$-$LCP(A, B, \epsilon)$, then it follows from the last two paragraphs that $\alpha \geq \alpha_{max}(\epsilon)$. However as already mentioned, since some of the alkyl residues of $\alpha_{max}(\epsilon)$-$LCP(A, B, \epsilon)$ might be outside, it is possible that they will be more than 8ϵ distance away from their corresponding points in B under any of the transformations tested by Algorithm 4.4. Under such circumstances α might be less than

$\alpha_{max}(\epsilon)$. We expect that for most practical problems $\alpha \geq \alpha_{max}(\epsilon)$ will hold, and even in cases where it does not, the difference will not be too large. Moreover, since any substructure identified by our scheme will ultimately be chemically evaluated to test its effectiveness, the substructure found by our algorithm should only be *large enough*. If A, B are point sets of cardinality $O(n)$, then transformations corresponding to $O(n^2)$ triplets are tested. The graph matching requires $O(n^{2.5})$ time. Hence the overall complexity of the algorithm is $O(n^{4.5})$.

4.4 An $O(n^{2.5}\log n)$ Randomized Approximation Algorithm for Proteins

In this section we give an algorithm, making use of the concepts presented in the last three sections. This algorithm is based on Algorithm 4.4, with the approximate graph matching algorithm $AM()$ being used to find the maximum matching in the bipartite graph corresponding to each transformation. Secondly, instead of computing transformations corresponding to all possible pairs of amino acids, we randomly sample a subset of amino acids from A and compute only those transformations corresponding to this subset, and the amino acids of B.

Algorithm 4.5

Input : *Labelled point sets A and B corresponding to two protein molecules and real numbers $\epsilon > 0, \delta > 0$.*

Output : *A real number α.*

A' = a randomly sampled multiset of k amino acids from A;
$M = 0$;
for each amino acid $a' \in A'$
  for each amino acid $b \in B$
  {
    $T_{a'b}$ = transformation that takes the N, C_α and C' atoms of a' to the corresponding atoms of b;

$M' = AM(T_{a'b}(A), B, 8\epsilon, \delta)$;
if $(M' > M)$ **then** $M = M'$;
}
return $M/\min(|A|, |B|)$;

Using the same reasoning as in Section 4.2, if the set A contains n amino acids, out of which atleast m belong to $\alpha_{max}(\epsilon)$-$LCP(A, B, \epsilon)$ then the set A' contains atleast one of them with probability $\geq q$, for $k \geq \lceil \frac{n}{m} \ln \frac{1}{1-q} \rceil$. If this happens then Algorithm 4.4 computes atleast one transformation which maps the skeleton corresponding to the common substructure of A into the corresponding skeleton of B. Hence Algorithm 4.4 returns with probability atleast q, a real number α such that there exists a subset $S \subseteq A$ of size $\alpha \min(|A|, |B|)$ which is $8\epsilon(1+\delta)$-congruent to some subset of B. Moreover, in all practical problems we expect α to be greater than, or atleast very close to $\alpha_{max}(\epsilon)$. Since the set A' is of cardinality $O(1)$, if there are n amino acids in the protein corresponding to the set B, then $O(n)$ transformations are computed. Graph matching corresponding to each transformation takes $O(n^{1.5} \log n)$ time. Hence the overall complexity of Algorithm 4.5 is $O(n^{2.5} \log n)$. The corresponding algorithm of Akutsu returns a subset $S \subseteq A$ which is 8ϵ-congruent to some subset of B and runs in time $O(n^8)$. However, it is guaranteed that $|S| \geq \alpha_{max}(\epsilon) \min(|A|, |B|)$ will hold.

It should be noted that instead of testing all the amino acids corresponding to the set B, we could have randomly sampled a subset of B in the same way as was done for set A. The resulting algorithm will have a time complexity of $O(n^{1.5} \log n)$. However, the success probability would change from q to $1 - e^{-\frac{cm}{n^2}}$, where c is an arbitrary constant.

Chapter 5

Conclusions

In this thesis we have presented several algorithms for identifying the structural similarities between two drug or protein molecules. An abstraction of this problem is that of finding the largest common point set under ϵ-congruence, between two point sets in 3-D. This has important applications in biomolecular recognition and binding, synthetic drug design, and molecular database screening. Most commonly used algorithms which address this problem are based on character string comparison algorithms, and hence in the case of proteins consider only the primary structure of the protein chain. But all our algorithms directly address the inherent three dimensional structure of the molecules, which is essential to find out any meaningful similarity between two molecules.

Although we have not presented any exact algorithm for the general problem, there is evidence that such an algorithm will have an exceedingly high running time. All our algorithms for the general isometry were approximation algorithms, with a guaranteed performance bound. Keeping in mind our primary application, given two molecules, since the allowable point location error ϵ is fixed by the chemist, our objective has been to find out the largest common substructure between the two molecules which conform to this allowable error. In our abstract problem this amounts to finding out $\epsilon_{max}(\epsilon)$. Towards this we have presented two classes of algorithms. The first approximates $\epsilon_{max}(\epsilon)$ by satisfying the point location error ϵ exactly, and the second approximately satisfies ϵ. In both the cases we provide

5.1 Future Research

Throughout this thesis we have treated all molecules to be rigid bodies, and considered only rigid transformations to superimpose one molecule on the other. Although such a treatment is adequate for comparing molecules with strong similarities, it will fail to identify weak similarities between pairs of molecules. So methods to overcome this need to be explored in future work.

It is not sufficient to identify the similarities only between pairs of molecules. Most applications require the identification of the common substructure in a group of molecules. For m collections of n points on the real line, the largest common point set cannot be approximated to within an n^{ϵ} factor unless $P = NP$, and only weak positive results are known [AH]. Hence it is expected that extending our algorithms to find out the largest common point set of m point sets in 3-D, will be exceedingly difficult. Although Finn *et al.* [FKL+97] have addressed this problem, no theoretical study was done.

One can view our algorithms only as a basic paradigm. We have shown that using a very general structural property of proteins, it is possible to improve their time complexities by a considerable extent. In a similar manner, additional biological information can be incorporated into this basic framework. This emphasizes the importance of an interdisciplinary research involving Biology, Chemistry, and Computer Science, in this area.

Bibliography

[ACM92] *Proc. 8th. Annual ACM Symp. on Computational Geometry*, 1992.

[ACM96] *Proc. 12th. Annual ACM Symp. on Computational Geometry*, Philadelphia PA, USA, 1996.

[ACM97] *Proc. 13th. Annual ACM Symp. on Computational Geometry*, Centre Universitaire Méditerranéen, Nice, France, 1997.

[AES95] P. K. Ararwal, A. Efrat, and M. Sharir. Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications. In *Proc. 11th. Annual ACM Symp. on Computational Geometry*, pages 39–50, 1995.

[AH] T. Akutsu and M. M. Halldórsson. On approximation of largest common subtrees and largest common point sets. LNCS 834. 405–413.

[AKM+92] E. M. Arkin, K. Kedem, J. S. B. Mitchell, J. Sprinzak, and M. Werman. Matching points into pairwise disjoint noise regions: Combinatorial bounds and algorithms. *ORSA J. Computing*, 4(4):375–386, 1992.

[Aku92] Tatsuya Akutsu. Algorithms for determining the geometrical congruity in two and three dimensions. In *Proc. 3rd. International Symposium on Algorithms and Computation*, Nagoya, Japan, December 1992. LNCS 650, pp. 279–288.

[Aku96] Tatsuya Akutsu. Protein structure alignment using dynamic programming and iterative improvement. *IEICE Trans. Inf. & Syst.*, E78-D(0), 1996.

[AMN+94] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Wu. An optimal algorithm for approximate nearest neighbor searching. In *Proc. 5th. Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 573–582, 1994.

[AMWW88] H. Alt, K. Mehlhorn, H. Wagener, and E. Welzl. Congruence, similarity, and symmetries of geometric objects. *Discrete and Computational Geometry*, 3:237–256, 1988.

[Ata84] M. J. Atallah. Checking similarity of planar figures. *Internat. J. Comput. Inform. Sci.*, 13:279–290, 1984.

[Atk87] M. D. Atkinson. An optimal algorithm for geometrical congruence. *J. Algorithms*, 8:159–172, 1987.

[ATT97] T. Akutsu, H. Tamaki, and T. Tokuyama. Distribution of distances and triangles in a point set and algorithms for computing the largest common point sets. In *Proc. 13th. Annual ACM Symp. on Computational Geometry* [ACM97], pages 314–323.

[Ber57] C. Berge. Two theorems in graph theory. *Proc. Natl. Acad. Sci. U.S.*, 43:842–844, 1957.

[BGSS] D. Barnum, J. Greene, A. Smellie, and P. Sprague. Identification of common functional components among molecules. *J. Chem. Inf. Comput. Sci.* To appear.

[Boy90] D. B. Boyd. Aspects of molecular modelling. In K. Lipkowitz and D. B. Boyd, editors, *Reviews in Computational Chemistry*, volume 1, pages 321–351. VCH Publishers, 1990.

[BT91] C. Braden and J. Tooze. *Introduction to Protein Structure.* Garland Publishing Inc., New York and London, 1991.

[CCC93] *Proc. 5th. Canadian Conference on Computational Geometry*, Waterloo, Canada, 1993.

[CGH+93] P. Chew, M. Goodrich, D. Huttenlocher, K. Kedem, J. Kleinberg, and D. Kravets. Geometric pattern matching under eucledian motion. In *Proc. 5th. Canadian Conference on Computational Geometry* [CCC93], pages 151–156.

[CK92] L. P. Chew and K. Kedem. Improvements on geometric pattern matching. In *Proc. 3rd. Scand. Workshop on Algorithm Theory*, 1992. LNCS 621, pp. 318–325.

[DG97] F. Dehne and K. Guimarães. Exact and approximate computational geometry solutions of un unrestricted point set stereo matching problem. *Information Processing Letters*, 64:107–114, 1997.

[Din70] E. A. Dinitz. Algorithm for solution of a problem of maximum flow in a network with power estimation. *Soviet Math Dokl.*, 11:248–264, 1970.

[dRL95] P. J. de Rezende and D. T. Lee. Point set pattern matching in d-dimensions. *Algorithmica*, 13:387–404, 1995.

[EI96] Alon Efrat and Alon Itai. Improvements on bottleneck matching and related problems using geometry. In *Proc. 12th. Annual ACM Symp. on Computational Geometry* [ACM96], pages 301–310.

[FBNW92] D. Fischer, O. Bachar, R. Nussiniv, and H. Wolfson. An efficient automated computer vision based technique for detection of three dimensional structural motifs in proteins. *Journal of Biomolecular Structures and Dynamics*, 9(4):769–789, 1992.

[FKL+97] P. W. Finn, L. E. Kavraki, J. C. Latombe, R. Motwani, C. Shelton, S. Venkatasubramanian, and A. Yao. RAPID: Randomized pharmacophore identification for drug design. In *Proc. 13th. Annual ACM Symp. on Computational Geometry* [ACM97], pages 324–333.

[FNW92] D. Fischer, R. Nussinov, and Hiam J. Wolfson. 3-d substructure matching in protein molecules. In *Proc. 3rd. Annual Symposium on Combinatorial Pattern Matching*, Tucson, Arizona, USA, April/May 1992. LNCS 644, pp. 136–150.

[GJ80] M. Garey and D. Johnson. *Computers and Intractability: A guide to the theory of NP-Completeness.* Freeman, San Francisco, 1980.

[GMO94] M. T. Goodrich, J. S. B. Mitchell, and M. W. Orletsky. Practical methods for approximate geometric patern matching under rigid motions. In *Proc. 10th. Annual ACM Symp. on Computational Geometry*, pages 103–112, 1994.

[Hef91] P. J. Heffernan. The translation square map and approximate congruence. *Information Processing Letters*, 39:153–159, 1991.

[HH92] J. E. Hopcroft and D. P. Huttenlocher. *Geometric Invariance on Computer Vision.* MIT Press, 1992. Chapter 18, pp. 354–374.

[HK73] J. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Computing*, 2:225–231, 1973.

[HK90] D. P. Huttenlocher and K. Kedem. Computing the minimum Hausdorff distance for point sets under translation. In *Proc. 6th. Annual ACM Symp. on Computational Geometry*, pages 340–349, 1990.

[HKK92] D. P. Huttenlocher, K. Kedem, and J. M. Kleinberg. On dynamic Voronoi diagrams and the minimum Hausdorff distance for point sets under Eucledian motion in the plane. In *Proc. 8th. Annual ACM Symp. on Computational Geometry* [ACM92], pages 110–120.

[HKS91] D. P. Huttenlocher, K. Kedem, and M. Sharir. The upper envelope of Voronoi surfaces and its applications. In *Proc. 7th. Annual ACM Symp. on Computational Geometry*, pages 194–203, 1991.

[HOS+92] L. Holm, C. Onzounis, C. Sander, G. Tuparev, and G. Vriend. A database of protein structure families with common folding motifs. *Protein Science*, 1:1691–1698, 1992.

[HS92] P. J. Heffernan and S. Schirra. Approximate decision algorithms for point set congruence. In *Proc. 8th. Annual ACM Symp. on Computational Geometry* [ACM92], pages 93–101.

[HU90] D. P. Huttenlocher and S. Ullman. Recognizing solid objects by alignment with an image. *International Journal of Computer Vision*, 5(2):195–212, 1990.

[IMV] P. Indyk, R. Motwani, and S. Venkatasubramanian. Geometric matching under noise: Combinatorial bounds and algorithms. Manuscript, July 5, 1997.

[IR96] S. Irani and P. Raghavan. Combinatorial and experimental results for randomized point matching algorithms. In *Proc. 12th. Annual ACM Symp. on Computational Geometry* [ACM96], pages 68–77.

[ISI89] K. Imai, S. Sumino, and H. Imai. Minimax geometric fitting of two corresponding sets of points. In *Proc. 5th. Annual ACM Symp. on Computational Geometry*, pages 276–282, 1989.

[ITY+] A. Itai, N. Tomioka, M. Yamada, A. Inoue, and Y. Kato. Molecular superposition for rational drug design. In Kubingi, editor, *3D QSAR in Drug Design: Theory, Methods and Applications.* ESCOM, 1995.

[Iwa90] S. Iwanowski. *Approximate congruence and symmetry detection in the plane.* PhD thesis, Fachbereich Mathematik, Freie Universität, Berlin, 1990.

[Lei43] Henry L. C. Leighton. *Solid Geometry and Spherical Trigonometry.* D. Van Nostrand Company Inc., Princeton, New Jersey, 1943. page 116.

[Les91] A. M. Lesk. *Protein Architecture: A practical approach.* IRL Press, New York, 1991.

[MARW89] E. M. Mitchell, P. J. Artymiuk, D. W. Rice, and P. Willet. Use of techniques derived from graph theory to compare secondary structure motifs in proteins. *Journal of Molecular Biology*, 212:151–166, 1989.

[MBD+93] Y. Martin, M. Bures, E. Danaher, J. DeLazzer, and I. Lico. A fast new approach to pharmacophore mapping and its application to dopaminergic and benzodiazepine agonists. *J. of Computer Aided Molecular Design*, 7:83–102, 1993.

[PA92] S. Pascarella and P. Argos. A data bank merging related protein structures and sequences. *Protein Engineering*, 5:121–137, 1992.

[PA94] Xavier Pennec and Nicholas Ayache. An $O(n^2)$ algorithm for 3d substructure matching for proteins. Technical Report N^o 2274, Unité de recherche INRIA Sophia Antipolis, 06902 Sophia Antipolis Cedex, France, Mai 1994.

[Rot91] G. Rote. Computing the minimum Hausdorff distance between two point sets on a line under translation. *Information Processing Letters*, 38:123–127, 1991.

[RR73] S. T. Rao and M. G. Rossmann. Comparison of super-secondary structures in proteins. *Journal of Molecular Biology*, 76:241–256, 1973.

[Ruc93] W. Rucklidge. Lower bounds for the complexity of the Hausdorff distance. In *Proc. 5th. Canadian Conference on Computational Geometry* [CCC93], pages 145–150.

[Sch88] Stefan Schirra. Über die Bitkomplexität der ϵ-Kongruenz. Diplomarbeit, 1988. Fachbereich Informatik, Universität des Saarlandes, Germany.

[Sch92] Stefan Schirra. Approximate decision algorithms for approximate congruence. *Information Processing Letters*, 43:29–34, 1992.

[ŠO94] A. Šali and J. P. Overington. Derivation of rules for comparative protein modelling from a database of protein structure alignments. *Protein Science*, 3:1582–1596, 1994.

[TO89] W. R. Taylor and C. A. Orengo. Protein structure alignment. *Journal of Molecular Biology*, 208:1–22, 1989.

[Vai89] P. M. Vaidya. Geometry helps in matching. *SIAM J. Computing*, 18:1201–1225, 1989.

[VS91] G. Vriend and C. Sander. Detection of common three-dimensional structures in proteins. *PROTEINS: Structure, Function and Genetics*, 11:52–58, 1991.

[Wil95] P. Willet. Searching for pharmacophoric patterns in databases of three dimensional chemical structures. *J. of Molecular Recognition*, 8:290–303, 1995.

[Wol90] H. Wolfson. Model-based object recognition by geometric hashing. In *Proc. 1st. European Conference on Computer Vision : ECCV90*, pages 526–536, 1990.